# A HYBRID TEMPLATE-BASED COMPOSITE CLASSIFICATION SYSTEM

DISSERTATION

Michael A. Turnbaugh, Captain, USAF

AFIT/DS/ENS/08-04

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

**AIR FORCE INSTITUTE OF TECHNOLOGY**

**Wright-Patterson Air Force Base, Ohio**

AFIT/DS/ENS/08-04

# A HYBRID TEMPLATE-BASED COMPOSITE CLASSIFICATION SYSTEM

DISSERTATION

Presented to the Faculty
Department of Operational Sciences
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

Michael A. Turnbaugh, B.S., M.S.
Captain, USAF

February 2009

AFIT/DS/ENS/08-04

A HYBRID TEMPLATE-BASED

COMPOSITE CLASSIFICATION SYSTEM

Michael A. Turnbaugh, B.S., M.S.

Captain, USAF

Approved:

[illegible] 6 FEB 2009

Dr. Kenneth W. Bauer Jr. Date
Committee Chair

[illegible] 6 Feb 2009

Dr. John O. Miller Date
Committee Member

Mark E. Oxley 6 Feb 2009

Dr. Mark E. Oxley Date
Committee Member

Accepted:

MU Thomas 6 Mar 2009

M.U. Thomas Date
Dean, Graduate School of Engineering
and Management

# Table of Contents

Page

Page

# List of Figures

Figure Page

# List of Tables

AFIT/DS/ENS/08-04

## Abstract

An automatic target classification system contains a classifier which reads a feature as an input and outputs a class label. Typically, the feature is a vector of real numbers. Other features can be non-numeric, such as a string of symbols or alphabets. One method of improving the performance of an automatic classification system is through combining two or more independent classifiers that are complementary in nature. Complementary classifiers are observed by finding an optimal method for partitioning the problem space. For example, the individual classifiers may operate to identify specific objects. Another method may be to use classifiers that operate on different features. We propose a design for a hybrid composite classification system, which exploits both real-numbered and non-numeric features with a template matching classification scheme. This composite classification system is made up of two independent classification systems.

These two independent classification systems, which receive input from two separate sensors are then combined over various fusion methods for the purpose of target identification.

By using these two separate classifiers, we explore conditions that allow the two techniques to be complementary in nature, thus improving the overall performance of the classification system. We examine various fusion techniques, in search of the technique that generates the best results. We investigate different parameter spaces and fusion rules on example problems to demonstrate our classification system. Our examples consider various application areas to help further demonstrate the utility of our classifier. Optimal classifier performance is obtained using a mathematical framework, which takes into account decision variables based on decision-maker preferences and/or engineering specifications, depending upon the classification problem at hand.

The first example problem is the handwritten digit recognition problem. The handwritten digits used in this application come from the Modified National Institute of Standards and Technology (MNIST) database. Many previous digit recognition methodologies have been tested on the MNIST database or on subsets of the database, making it an excellent baseline for comparison of methods. The examination of this application demonstrates the versatility of the composite classification system and the framework used to optimize classifier performance.

The second application examined is a combat identification problem. The ability of a decision maker to make a quality real-time decision requires reliable and timely information must be made available. Within the scope of combat identification, this means combat identification systems must be fast, accurate and easy to use. We apply our composite classification system to a Synthetic Aperture Radar system data set, which was collected at Eglin AFB, FL from the General Dynamics Data Collection System. We examine methodologies for classification, fusion, non-declarations and out-of-library determination along with the mathematical framework to realize optimal classification system results which outperm previous research on this problem.

The results of this research are a novel out-of-library detector, which successfully identifies targets for which the classification system has not been trained to make decisions. This is the first such successful classifier of this type. This research also sucessfully combines elements of statistical template methods and syntactic methods to create a hybrid classification system that is both robust and fast, in terms of computation time, in both classification problems. We also advance the notion of Automatic Target Recognition as an engineering optimization problem and use a statistical model to demonstrate the best fusion optimization scheme.

## Acknowledgements

There are many people to whom I owe a great deal of thanks for their part in helping me sucessfully complete this program of study.

I would first like to thank my research advisor, Dr. Kenneth Bauer. His extradordinary vision, guidance, patience and experience was immeasurable. I have truly enjoyed working for Dr. Bauer and his ability to mentor and challenge his students made my time at AFIT remarkable.

I am also grateful to my committee, Dr. J.O. Miller and Dr. Mark Oxley. I thank them both for their guidance and feedback on my dissertation.

I must also thank several of my peers. Capt Nate Leap, Mr Scott Percival, Maj Mark Friend, Maj Joe Price and Maj June Rodriguez whose frienship and cooperation throughout this entire process kept me on track in every way. I truly enjoyed our time together through coursework and each of our own research endeavors. I will always treasure the bonds of friendship and professional collaboration we developed here. I couldn't have asked for a better group to share this experience with.

I must also thank my "crew", K., J. and M. whose friendship during my time here is immeasurable.

I thank my children for allowing me this time to pursue my studies. There were certainly times when they would have preferred that my time was free for them and I appreciate their patience, understanding and love during my time here. I also must say thank you to my parents. They both taught me to demand the best in myself, which served me well time and time again during my studies.

Finally, and most importantly, I need to thank my Lord and Savior, Jesus Christ, who has blessed me in so many ways. He has picked me up at times I didn't think possible. I find so much comfort in knowing he is in charge.

Michael A. Turnbaugh

A HYBRID TEMPLATE-BASED
COMPOSITE CLASSIFICATION SYSTEM

# 1. Introduction

## *1.1 Pattern Recognition and Hybrid Techniques*

The problem of pattern classification is approached from several different points of view. Jain *et al.* [25] list the following as the four best known approaches. The first is template matching. In template matching, prototypes of all known patterns are stored into a template, which is used for comparison to unknown patterns for classification. The classification decision in template matching is based on a similarity measure. These measures can include correlation or Mahalanobis distance. The second approach is the statistical approach. In this approach, each pattern is represented in terms of a number of features or measurements, which can then be viewed as a point in a $d$-dimensional vector space. The goal in the statistical method is to choose features such that the pattern vectors occupy disjoint spaces in the $d$-dimensional vector space. The third approach is neural networks. Neural networks can be described as massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. The fourth and last approach is the syntactic approach. The premise of the syntactic approach is that problems with complex patterns should adapt a hierarchical prospective in which each pattern is considered to be made up of subpatterns, which, in turn, are made up of smaller subpatterns. The smallest of these subpatterns are called primitives. From these primitives, the more complex patterns are represented in terms of how the primitives relate to one another. The analogy Jain uses is the comparison between the structure of patterns and the syntax of a language.

Bunke [7] describes the traditional methods of pattern recognition as being either: (1) decision theoretic or statistical; or (2) structural. The author goes on to explain that each of these different methods has its strengths and limitations. A methodology to combine multiple pattern recognition in such a way as to overcome the drawbacks of each while maintaining the advantages of each is known as a hybrid pattern recognition system [14].

We choose to call our technique a hybrid to reflect that our classifier combines sereral different techniques. The non-numeric features are from syntactic techniques. The formulation of templates and the use of geometric type distances are from the template-based techniques. The use of thresholds is from statistical techniques.

It can be argued that any pattern recognition system can be described as a hybrid, rather than belonging to a particular class or approach, such as those listed by Jain. The main objective in this research is to build a hybrid classification system, which combines strengths from several different known pattern recognition techniques, that can be adapted such that it can be successfully applied across different application areas.

## *1.2 Combat Identification and Automatic Target Recognition*

The ability of a decision maker to make a quality real-time decision requires that reliable and timely information must be made available. Within the scope of combat identification, this means combat identification systems must be fast, accurate and easy to use. We now demonstrate how our classification system makes improvements to the timeliness and accuracy of combat identification systems by automatic target recognition methods. The process of detecting, tracking and correctly identifying an enemy's key targets loosely defines the combat identification (CID) process.

Sadowski [57] defined CID as "the process of attaining an accurate characterization of detected objects in the joint battlespace to the extent that high confidence, timely application of tactical military options and weapons resources can occur."

Laine [34] describes two types of CID: (1) cooperative and (2) non-cooperative. Cooperative CID includes identification of friendly targets through the use of communication between two friendly systems. Non-cooperative CID includes cases where feedback from one of the systems does not occur. This can be further broken up into fully autonomous or man-in-the-loop systems. In an autonomous system, decisions are made without any type of human intervention. On the other hand, man-in-the-loop systems require a human to make final decisions regarding target identification.

Automatic target recognition systems seek to fully automatate the target recognition process. This includes not only the recognition and descrimination between hostile or friendly targets, but also the ability to correctly categorize hostile targets for the purpose of engagement.

## *1.3 Research Goals and Application Areas*

The goal of this research is the development of a hybrid composite classification system, which exploits integer valued quantized features with a template matching classification scheme. This composite classification system is made up of two independent classification systems.

We investigate different parameter spaces and fusion rules on example problems to demonstrate our classification system. Our examples consider various application areas to help further demonstrate the utility of our classifier. Optimal classifier performance is obtained using a mathematical framework, which takes into account decision variables based on decision-maker preferences and/or engineering specifications, depending upon the classification problem at hand.

## *1.4 Contributions of Research*

This research makes several contributions within the overall research goal. First, is the hybrid template-based classifier we develop. We demonstrate the utility of the hybrid classifier across different problem types. Noting that we achieve superior or equal classifier performance to existing systems, while developing a system that has potential real-world application, due to lower computational and storage requirements. The development of this classifier includes the exploration of feature extraction, representation scheme, similarity measure, classification technique and fusion method in order to produce a flexible, optimal peforming system.

In the course of this research, we detail the mathematics of our classification system. In doing so, we produce a combined system that composes the hybrid classifier with an out-of-library OOL detector. This OOL detector uses artificial neural networks as a means of identifying targets (OOL targets) for which the hybrid classifier is not trained to recognize. We go on to develop an overall mathematical framework, that enables us to find optimal parameter settings for the overall classification system that optimize some measure of performance for the system.

## *1.5 Organization of Dissertation*

This disseration is organized as follows. Chapter 2 provides a thorough background via a summary of current literature. We discuss various pattern recognition techniques, similarity measures and fusion techniques that may be used for the purpose of constructing a classification system. We also present a background of our two application areas. The first application area is target identification using High Range Resolution (HRR) profiles that have been extracted from Synthetic Apperture Radar (SAR) imagery. The second application area is recognition of optical characters. In Chapter 3, we provide the mathematical framework for our classification system. We present our methodology for the case of OOL targets as well as the case where the classification system is unable to distinguish between two or more potential target labeling options. Instead of making an uncertain decision, the classification system uses the labeling option of non-declaration (NDEC). Finally, we formulate a mixed variable optimization problem as well as various methods of evaluation in seeking to improve classifier performance. In chapter 4, we apply our classifciation system to the optical character recognition problem, and make positive comparisons to existing techniques. We apply our classification system to the ATR problem problem in Chapter 5, where we use the HRR profiles derived from SAR imagery. We apply our system in three different scenarios: a forced decision, a 10-class problem with a NDEC option and a problem that includes both in-library and out-of-library targets. Throughout this process, we develop methods and application techniques for both a NDEC labeling option and an OOL detector. Finally, we draw superior results to previous and/or exisiting techniques that have been applied to this problem through the use of the mathematic framework. Finally, Chapter 6 summarizes the contributions of this research and suggests areas for further research.

# 2. Background

## *2.1 Introduction*

This chapter reviews literature and provides background on concepts and methodologies related to research presented in subsequent chapters. First, a background of pattern recognition and various pattern recognition techniques is presented. Current application areas are discussed for each of the techniques and relevant literature reviews are presented. Second, similarity measures used in pattern recognition are discussed and our specific choice of measure is presented. Third, various data fusion techniques are presented with relevant literature for each. Next, we present background for each of the two application areas which will be presented in subsequent chapters of this document. First, we discuss the formation of high-range resolution (HRR) profiles. The data set used in our research is presented, along with the preprocessing steps which produce the HRR profiles. In addition, the current literature for HRR profile classification is presented. Finally, handwritten character recognition techniques are discussed and the current literature is reviewed. Along with this discussion, we detail the handwritten character data set used in our research.

## *2.2 Pattern Recognition Background*

The problem of pattern classification is approached from several different points of view. Jain *et al.* [25] list the following as the four best known approaches, with brief descriptions of each taken from this paper. The first is template matching. In this approach, prototypes of all known patterns are stored into a template, which is used for comparison to unknown patterns for classification. The classification decision in template matching is based on a similarity measure. These measures can include correlation or Mahalanobis distance [3, 12, 34]. The second approach is the statistical approach. In this approach, each pattern is represented in terms of a number of features or measurements, which can then be viewed as a point in a $d$-dimensional vector space. The goal in the statistical method is to choose features such that the pattern vectors occupy disjoint spaces in the $d$-dimensional vector space. The third approach is neural networks. Jain describes neural networks as massively parallel computing systems consisting of an extremely large number of simple processors with many interconnections. The fourth and last approach we will mention is the syntactic approach. The premise of the syntactic approach is that problems with complex patterns should adapt a hierarchical prospective in which each pattern is considered to be made up of subpatterns, which in turn are made up of smaller subpatterns. The smallest of these subpatterns are called primitives. From these primitives, the more complex patterns are represented in terms of how the primitives relate to one another. The obvious analogy Jain uses is the comparison between the structure of patterns and the syntax of a language.

### *2.2.1 Statistical Techniques*

As previously stated, Jain *et al.* [25] state that in the statistical approach to pattern recognition, each pattern is represented in terms of a number of features or measurements, which can then be viewed as a point in a $d$-dimensional vector space, $\mathbf{x} = (x_1, x_2, ..., x_d)$. The goal in the statistical method is to choose features such that the pattern vectors occupy disjoint sets in the $d$-dimensional vector space. In designing a statistical classifier, it is assumed that there exist $C$ classes denoted $\omega_1, \omega_2, ..., \omega_C$, and associated with each pattern $\mathbf{x}$ is a categorical variable, $z$ that denotes class membership. Thus, if $z = i$, then the pattern belongs to class $\omega_i$, $i \in 1, 2, ..., C$ [69]. Webb explains that class membership determinations are made through a decision rule, which partitions the feature space into $C$ regions $\Omega_i$, $i = 1, 2, ..., C$. If an exemplar, $\mathbf{x} \in \Omega_i$, then it is assigned to class $\omega_i$ [69]. Discrimination can generally be broken down into two approaches. The first assumes knowledge of the underlying class-conditional probability density functions of the feature vectors, while the second approach makes decision rules based on the data itself without making assumptions about or calculating the probability density functions.

#### *2.2.1.1 Bayesian Decision Theory*

Consider $C$ classes $\omega_1, \omega_2, ..., \omega_C$, with *a priori* probabilities $P(\omega_1), P(\omega_2), ..., P(\omega_C)$, which are assumed to be known. If a classification decision must be made about observation vector $x$ with only this information, then we decide $x \in \omega_i$ if

$$P(\omega_i) > P(\omega_j) \qquad j = 1..., C; \quad j \neq i \tag{2.1}$$

If we consider the observation vector, $\mathbf{x} = (x_1, x_2, ..., x_d)$, we can base our decision on class-conditional probabilities, $p(\omega_i|\mathbf{x})$. Thus, we assign $\mathbf{x}$ to class $\omega_i$ if

$$p(\omega_i|\mathbf{x}) > p(\omega_j|\mathbf{x}) \qquad j = 1, ..., C; \quad j \neq i \tag{2.2}$$

The *a posteriori* probabilities, $p(\mathbf{x}|\omega_i|)$ may be expressed in terms of the *a priori* probabilities $P(\omega_i)$ and the class-conditional density functions $p(\omega_i|\mathbf{x})$ using Bayes' formula

$$p(\mathbf{x}|\omega_i) = \frac{p(\omega_i|\mathbf{x})P(\omega_i)}{p(\mathbf{x})} \tag{2.3}$$

where the evidence from the data is

$$p(\mathbf{x}) = \sum_{i=1}^{C} p(\mathbf{x}|\omega_i)p(\omega_i).$$

Thus, the decision rule can be written: assign $\mathbf{x}$ to $\omega_i$ if

$$p(\omega_i|\mathbf{x})p(\omega_i) > p(\omega_j|\mathbf{x})p(\omega_j) \qquad j = 1, ..., C; \quad j \neq i \tag{2.4}$$

This is known as Bayes' rule for minimizing the probability of a classification decision error [11,69]. Both Webb and Duda *et al.* dedicate several chapters of each textbook detailing Bayesian decision theory [11,69].

#### *2.2.1.2 Discriminant Functions*

Webb [69] introduces the technique of discriminant functions as a method to complement Bayesian decision theory. With this method, assumptions about the forms of the discriminant functions, $g_i(\mathbf{x}), i = 1, 2, ..., C$ are made, rather than making assumptions about $p(\mathbf{x}|\omega_i)$. Duda *et al.* [11] explains that a classifier will assign an exemplar $\mathbf{x}$ to class $\omega_i$ if

$$g_i(\mathbf{x}) > g_j(\mathbf{x}) \qquad j = 1, ..., C; \quad j \neq i \tag{2.5}$$

Webb [69] details several examples of discriminant functions, which are shown in Table 2.1.

Example application areas for statistical pattern recognition techniques include the following. Facial recognition is an ongoing research effort that employs statistical pattern recognition techniques. Bayesian methods using a probablistic distance met-

Table 2.1 Discriminant Functions, $\phi$

| Discriminant Function | Mathematical Form, $\phi_i(\mathbf{x})$ |
|---|---|
| linear | $\phi_i(\mathbf{x}) = x_i, \quad i = 1, ..., p$ |
| quadratic | $\phi_i(\mathbf{x}) = x_{k_1}^{l_1} x_{k_2}^{l_2}, \quad i = 1, ..., \frac{(p+1)(p+2)}{2-1}$ |
| | $l_1, l_2 = 0$ or $1$; $k_1, k_2 = 1, ..., p$; $l_1, l_2$ not both 0. |
| $\nu$th order polynomial | $\phi_i(\mathbf{x}) = x_{k_1}^{l_1} ... x_{k_\nu}^{l_\nu}, \quad i = 1, ..., \binom{p+\nu}{\nu} - 1$ |
| | $l_1, ..., l_\nu = 0$ or $1$; $k_1, ...k_\nu = 1, ..., p$; $l_i$ not all 0 |
| radial basis function | $\phi_i(\mathbf{x}) = \phi(\lvert\mathbf{x} - \nu_\mathbf{i}\rvert)$ for center $\nu_i$ and function $\phi$ |
| multilayer perceptron | $\phi_i(\mathbf{x}) = f(\mathbf{x}^\mathbf{T}\nu_\mathbf{i} + \nu_{i0})$ for direction $\nu_\mathbf{i}$ and offset $\nu_{i0}$. |
| | $f$ is the logistic function, $f(z) = 1/(1 + exp(-z))$ |

ric as a decision criteria were employed in work by Moghaddam and Pentland [47]. Statistical techniques have also been employed in the area of handwritten character recognition [?] and automatic target recognition [45].

### *2.2.2 Syntactic Techniques*

To begin, we will consider some of the elementary basics of Syntactic Pattern Recognition. We will then explore some of the current research being conducted which used the syntactic approach to pattern recognition. Fu [14] describes a syntactic pattern recognition system as being made up of two major parts: analysis and recognition. The analysis part consists of selecting the primitives and grammatical inference. Grammars or syntax rules are the means to describe the rules of languages or the structural relations of patterns. Fu defines the problem of learning a grammar based on a set of sample sentences as grammatical inference. Thus, our goal in syntactic pattern recognition is to choose primitives that are adequately representative of the subpatterns that make up the more complex patterns in our approach to classification. We then form grammars which will tell us the rules for putting together the simple primitives to form more complex patterns. Again, we can use the direct analogy of how words, sentences and paragraphs are formed from the grammatical rules of a language to the way in which primitives can be used to form more complex patterns by way of the grammatical rules employed.

There are several current research areas which employ the syntactic approach. The medical community has tapped into the use of syntactic pattern recognition for aid in diagnosis of cancer such as Ogiela [49] as well as in the aid of identification of abnormal electro-cardiograms done by Trahanias [61]. In the former, a context-free attribute grammar was developed to symbolically depict the characteristic of a pancreas as seen on an x-ray image. The primitives of the grammar were chosen in a way such that representations would note significant changes, such as cysts, branchings or enlargements; all of which are indicative of pancreatic cancer. In the latter case, syntactic pattern recognition techniques are applied to electrocardiograms so that an automated process for detection of abnormal readings could be applied. The primitives chosen for this application were chosen to represent the key elements of the electrocardiogram, the complexes. Each of these complexes contains either parabolic shapes, sharp peaks or line segments as well as combinations of the three. Thus the primitives were chosen to be these three distinct attributes, with measurements of each, such as starting point, stopping point and peak amplitude captured as part of the overall representations.

Other application areas include character recognition, RNA modeling and imagery analysis. Particular areas of character recognition include recognition of Chinese characters. In the work by Liu *et al.* [39], the particular goal of online handwriting recognition of handwritten Chinese characters is addressed. The increased interest in this field is due to the increase of pen computing devices and pen input devices. They use a representation scheme that accounts for the relationships between the complex strokes of these Chinese characters, followed by a template matching type scheme that matches candidate classes to a library of known classes. Another application to Chinese character recognition is done by Kuroda [33], which employs a Kohonen self-organizing map as a method of feature extraction. The results in their experiments show promise, with a recognition rate of over 94 percent. In the field of RNA modeling, Abe [2] used grammatical inference by way of a Hidden Markov Model algorithm to automatically learn the RNA sequences. The protein secondary structures of an RNA sequence were predicted using grammatical inference by Sakakibara [58]. Image analysis of Synthetic Aperture Radar ship images using syntactic methods was done by Klepko [30]. Syntactic methods can also be applied to the analysis and target identification of images captured by space borne platforms [24] [38].

The non-numeric features of the hybrid classifier developed in this research are inspired by the syntactic classifier. The hybrid classifier attempts to represent a given signal using this scheme such that a new representation unlike those found in previous research is formed. This new classifier can then in turn be fused with other classifiers that use a complementary representation scheme to produce the combined classification system that shows improved overall performance.

### *2.2.3 Template-based Techniques*

The feature vector classifier used in this research is a slight modification from Friend [12]. This research conducted ATR on HRR signatures derived from SAR chips using algorithms provided by AFRL, as shown in Figure 2.1. These 724 HRR profiles are then sorted by aspect angle, evenly divided into training and test sets then interpolated to produce two $360°$ data sets for each target.

In the training phase, a 10-dimensional feature vector is extracted by binning the HRR profile data into 10 equally sized range bins, as shown in Figure 2.2. The features are the maximum amplitude within each bin. Templates are then formed for each target, where each template is made up of a feature vector at each aspect angle. In the classification phase, an unknown HRR profile is compared to the template in the following manner. Twenty-four wedges of $15°$ width are made from each target template. Using a prior aspect angle knowledge of $\pm 22.5°$, test profiles are compared to template profiles from surrounding two wedges for each target. For example, a test profile with an aspect angle of $20°$ would be compared to wedges 1-3, or a template window of size $45°$. The minimum squared Mahalanobis distance over all three wedges is then used for comparison between the unknown profile and

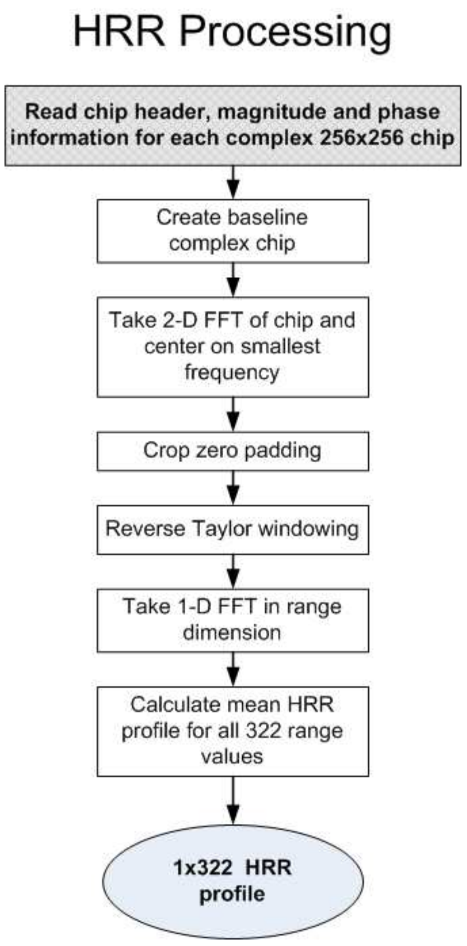
HRR Processing
Read chip header, magnitude and phase information for each complex 256x256 chip
Create baseline complex chip
Take 2-D FFT of chip and center on smallest frequency
Crop zero padding
Reverse Taylor windowing
Take 1-D FFT in range dimension
Calculate mean HRR profile for all 322 range values
1x322 HRR profile

Figure 2.1 Steps to process SAR chips into HRR profiles.

each of the possible classes. This research also considered different prior aspect angle knowledge, which is discussed in subsequent sections.

This method of choosing peak amplitudes within range bins was based on research by Mitchell and Westerkamp [45] who applied features from range bins with a statistical feature based classifier which was applied to HRR signatures. This statistical classifier introduced in Mitchell [46] used features extracted exclusively from the middle portion of the signal, which the author states is the portion of the signal which contains useful information. This research also used the technique of using the

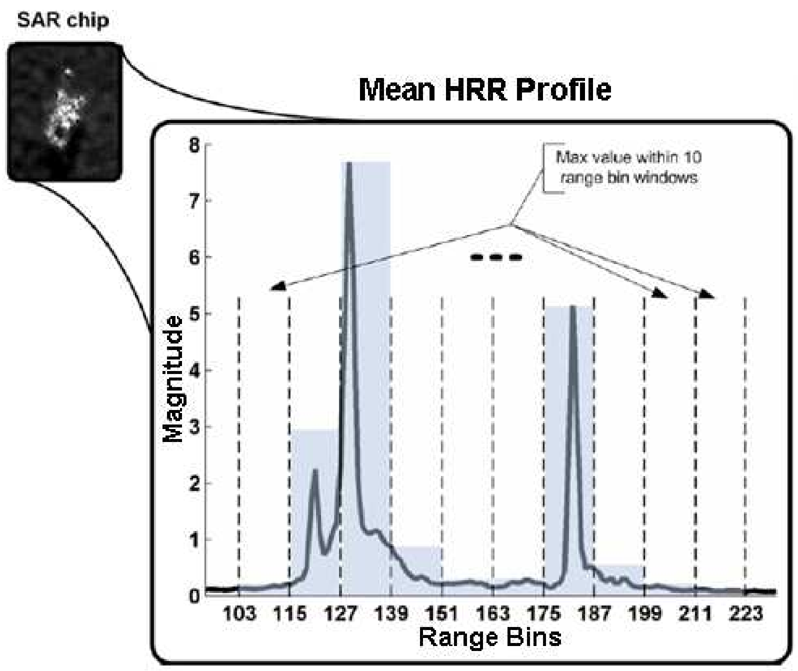

Figure 2.2  Example Range Bins used by Feature Vector Classifier.

peak amplitudes of an HRR profile as features for representation. Another feature used in representation was peak location, which naturally aids in registration. The classification decisions are then based on two models generated from the features extracted; a peak location probability function and peak amplitude probability density function. The experimental results include forced decision as well as forced decision with unknown target types, which we call out-of-library targets.

The Multinomial Pattern Matching algorithm developed by Sandia National Laboratories [32] is another template-based classification method that has been successfully applied to HRR signature classification. The MPM utilizes a quantile transformation to map target intensity samples to a small number of grayscale values, or quantiles. The MPM builds a template of HRR signatures in the training phase, which is described in the following paragraphs.

The data used in their research contains multiple signatures for a given aspect angle. Each HRR signature is mapped to a fingerprint using the quantization method. These profile fingerprints are then binned by a given aspect angle, which is chosen to be $10°$ in this paper. The bin width is chosen such that it is wide enough to populate aspect bins over even a short tracking engagement yet narrow enough to preclude drastic changes in target signature over the width of the aspect bin.

Profile stabilization is performed to maximize the similarity between profiles within any given aspect bin. This profile stabilization consists of profile alignment followed by optional length normalization and smoothing steps. For narrow aspect bins, the alignment process generally yields acceptable registration. For wider aspect bins, the length normalization is implemented to aid in the registration of profiles with varying target lengths.

The target templates are then formed consisting of two components. The first is a statistical characterization component that forms a marginal quantile model as a $K \times N_q$ matrix of observed quantile probabilities $\hat{\mathbf{P}}$. The second is a $K \times N_q$ matrix of sample penalties $\mathbf{T}$, which expresses the penalty to be assigned to any quantile observation at any sample. In each case, $K$ is the number of samples and $N_q$ is the the number of quantiles. These two matrices are then used in template matching by computing a z-score for any unclassified exemplar when compared to a given target template.

Another template-based approach to the HRR signature matching problem was done by Bhatnagar *et al.* [5]. This research developed a structural approach for HRR pattern recognition. Their technique used grammatical inference and classification algorithms, attribute grammars and an error correcting parsing mechanism. With this method, HRR patterns were classified using structural as well as quantitative information from the numerical attributes of the pattern grammars.

Classification decision were done using a minimum distance classifier based on syntactic approach. The choice of weighed Levenshtein distance measure was used for comparing the distance between string representations of test profiles with those of a template formed in the training phase.

### *2.2.4 Hybrid Techniques*

Bunke [7] describes the traditional methods of pattern recognition as being either: (1) decision theoretic or statistical; or (2) structural. The author goes on to explain that each of these different methods has its strengths and limitations. A methodology to combine multiple pattern recognition in such a way as to overcome for the drawbacks of each while maintaining the advantages of each is known as a hybrid pattern recognition system [13].

Nadler [48] describes pattern recognition in the following manner. The task of removing noise and revealing the underlying ideal pattern. In the formation of a hybrid method, he states it is advantageous to pattern recognition problems to use a combined method. He also states the desired outcome of combining methods is to overcome the weaknesses of each method while using the strengths of each. Before moving on, let us consider the strengths and weaknesses of each approach.

The statistical approach has a very rich and well established theoretical foundation. It has given rise to numerous proven methods that have been applied to applications both within and beyond pattern recognition. Statistical methods are most useful when the number of prototypes is relatively small and the range of variation from each prototype is small enough that clusters of classes do not overlap [48]. One weakness of the statistical approach is the feature extractor. No common theory exists for the design and selection of optimal features. Optimal feature extraction is typically the result of cleverness and experience of the designer.

Syntactic techniques do not rely on features. Instead, as we have seen these approaches attempt to learn a grammar through inference that builds a language of words or representations for the set of prototypes. These methods do not need the measurements counted on by statistical approaches, thus the overlapping of prototypes is not as destructive as in statistical methods. One drawback of the syntactic approach is the computational efficiency. The number of distinct words that can arise from a single class can be extremely large [14]. This is due to the fact that small imperfections in shape that would be ignored by a statistical classifier can generate entire new characters in the syntactic approach.

Template methods store prototypes of all known patterns and use these prototypes for comparison to unknown patterns. These methods are not as theoretically sound as the previous methods. Furthermore, the number of prototypes needed to accurately depict all known or observed patterns can be extremely high. These template approaches use some form of similarity measure as a means for classification. With the possibly large number of prototypes, this gives rise for the need to use centroids such as the mean or other computed representation as a prototype when building a class template.

We choose to call our technique a hybrid to reflect that this classifier combines several different techniques. The non-numeric features are from syntactic techniques. The formulation of templates and the use of geometric type distances are from the template-based techniques. The use of thresholds is from statistical techniques.

## *2.3 Similarity Measures*

The concept of similarity is fundamental to pattern recognition systems. Classification decisions are routinely based on the level of similarity a given input pattern has to a template of prototypes, a class distribution or some other prototype which is used as the basis of a given class of patterns. One of the oldest and most influential similarity concepts is that perceived similarity is inversely related to observed distance. This means that if we take some form of distance measure between two objects, the closer they are in distance implies the more similar the two objects. This section will give a brief overview of some of the methods used in determining similarity.

### *2.3.1 Hamming Distance*

In the theory of block codes intended for error detection or error correction, the Hamming distance $d(u, v)$ between two words $u = (u_1, u_2, ..., u_n)$ and $v = (v_1, v_2, ..., v_n)$, of the same length, is equal to the number of symbol places in which the words differ from one another. If $u$ and $v$ are of finite length $n$ then their Hamming distance is finite since $d(u, v) \leq n$ [16].

The Hamming distance can be called a distance since it is nonnegative, positive definite, symmetric, and triangular:

$$
\begin{aligned}
\text{Nonnegative:} \quad d(u, v) &\geq 0 && (2.6)\\
\text{Positive Definite:} \quad d(u, v) &= 0 \quad \text{iff} \quad u = v && (2.7)\\
\text{Symmetric:} \quad d(u, v) &= d(v, u) && (2.8)\\
\text{Triangular:} \quad d(u, w) &\leq d(u, v) + d(v, w) && (2.9)
\end{aligned}
$$

The Hamming distance is important in the theory of error-correcting codes and error-detecting codes: if, in a block code, the codewords are at a minimum Hamming distance $d$ from one another, then:

(a) if $d$ is even, the code can detect $d-1$ symbols in error and can correct $\frac{1}{2}d-1$ symbols in error;

(b) if $d$ is odd, the code can detect $d-1$ symbols in error and can correct $\frac{1}{2}(d-1)$ symbols in error.

### *2.3.2 Levenshtein Distance*

Levenshtein distance ($LD$) is a measure of the similarity between two strings, which we will refer to as the source string $s$ and the target string $t$. The distance is the number of deletions, insertions, or substitutions required to transform $s$ into $t$ [18]. For example,

If $s$ is the string $s=(t,e,s,t)$ and $t$ is the string $t=(t,e,s,t)$, then $LD(s,t)=0$, because no transformations are needed. The strings are already identical. If $s$ is the string $s=(t,e,s,t)$ and $t$ is the string $t=(t,e,n,t)$, then $LD(s,t)=1$, because one substitution (change "s" to "n") is sufficient to transform $s$ into $t$. The greater the Levenshtein distance, the more dissimilar the strings are.

The Levenshtein distance algorithm has been used in such applications as spell checking, speech recognition, DNA analysis and plagiarism detection.

The Levenshtein distance also has several simple upper and lower bounds that are useful in applications which compute many of them and compare them. These include:

1. It is always at least the difference of the sizes of the two strings,

$$LD(s,t) = \min\left\{|s|, |t|\right\}.$$

2. It is at most the length of the longer string,

$$LD(s,t) \leq \max\left\{|s|, |t|\right\}.$$

3. It is zero if and only if the strings are identical,

$$LD(s,t) = 0, \quad \text{iff} \quad s = t.$$

4. If the strings are the same size, the Hamming distance $d(s,t)$ is an upper bound on the Levenshtein distance

$$LD(s,t) \leq d(s,t) \quad \text{iff} \quad |s| = |t|.$$

### *2.3.3 Jaro Winkler Algorithm*

The Jaro-Winkler algorithem [74] measures the similarity between two strings. It is a variant of the Jaro algorithm [26] and mainly used in the area of record linkage (duplicate detection). The higher the Jaro-Winkler score for two strings is, the more similar the strings are. The Jaro-Winkler algorithm is designed and best suited for short strings such as person names. The score is normalized such that 0 equates to no similarity and 1 is an exact match.

The Jaro algorithm states that given two strings $s_1$ and $s_2$, their similarity $d_J$ is:

$$d_J = \frac{1}{3}\left(\frac{m}{\|s_1\|} + \frac{m}{\|s_2\|} + \frac{m-t}{m}\right) \tag{2.10}$$

where:

$m$ is the number of matching characters and

$t$ is the number of transpositions.

Two characters from $s_1$ and $s_2$ respectively, are considered matching only if their distance $d_J$ is not greater than:

$$\lfloor \frac{\max(\|s_1\|, \|s_2\|)}{2} \rfloor - 1$$

Each character of $s_1$ is compared with all its matching characters in $s_2$. The number of matching (but different order) characters divided by two defines the number of transpositions.

Jaro-Winkler algorithm uses a prefix scale $p$ which gives more favorable ratings to strings that match from the beginning for a set prefix length,$l_p$. Given two strings $s_1$ and $s_2$, their Jaro-Winkler distance $d_JW$ is:

$$d_{JW} = d_J + l_p(1 - d_J) \tag{2.11}$$

where:

$d_J$ is the Jaro distance for strings $s_1$ and $s_2$,

$l_p$ is the length of common prefix at the start of the string up to a maximum of 4 characters and

$p$ is a constant scaling factor for how much the score is adjusted upwards for having common prefixes.

The standard value for this constant in Winkler's work is $p = 0.1$. Although often referred to as a distance metric, the JaroWinkler score is actually not a metric in the mathematical sense of that term because it fails the triangle inequality.

For example, given the strings $s_1 = (t, a, w, n, y, a)$ and $s_2 = (t, o, n, y, a)$, we find

$m = 3$

$\|s_1\| = 6$

$\|s_2\| = 5$

$t = 0$

then the Jaro score for the strings is:

$d_J = \frac{1}{3}\left(\frac{3}{6} + \frac{3}{5} + \frac{3}{3}\right) = 0.7.$

To find the Jaro-Winkler score, we find

$l_p = 1.$

Thus, the Jaro-Windler score is: $d_{JW} = 0.70 + (1 * 0.1(1 - 0.70)) = 0.73.$

### *2.3.4 Minkowski Metric*

In the Euclidean space $\mathbf{R}^n$, the distance between two points is usually given by the Euclidean distance (2-norm distance) [11]. Other distances, based on other norms, are sometimes used instead.

For a point $\mathbf{x} = (x_1, x_2, ..., x_n)$ and a point $\mathbf{y} = (y_1, y_2, ..., y_n)$, the Minkowski distance of order $p$ ($p$-norm distance) is defined as:

$$d_p(\mathbf{x}, \mathbf{y}) = \|x - y\| = \left(\sum_{k=1}^{n} |x_k - y_i|^p\right)^{1/p} \tag{2.12}$$

the order, $p$ need not be an integer, but it cannot be less than 1, because otherwise the triangle inequality does not hold [11].

The 2-norm distance is the Euclidean distance, $d_2$, a generalization of the Pythagorean theorem to more than two coordinates. It is what would be obtained if the distance between two points were measured with a ruler: the "intuitive" idea of distance.

The 1-norm distance is called the taxicab norm or Manhattan distance, $d_1$, because it is the distance a car would drive in a city laid out in square blocks (if there are no one-way streets).

The infinity norm , $\|x\|_\infty \equiv \max_{i \in 1,\ldots,n} \|x_i\|$ is also called Chebyshev distance [27]. In $\mathbf{R}^2$ it represents the distance kings must travel between two squares on a chessboard.

The $p$-norm is rarely used for values of $p$ other than 1, 2, and infinity.

In physical space the Euclidean distance is, in a way, the most natural one, because in this case the length of a rigid body does not change with rotation.

### *2.3.5 Cosine Similarity*

Cosine similarity is a measure of similarity between two vectors of $n$ dimensions by finding the angle between them, often used to compare documents in text mining. Given two vectors, $\mathbf{x} = (x_1, x_2, ..., x_n)$ and $\mathbf{y} = (y_1, y_2, ..., y_n)$, such that

$\mathbf{x} \neq \mathbf{0}$ and $\mathbf{y} \neq \mathbf{0}$, the cosine similarity, $\theta$, is represented using a dot product and magnitude as:

$$\theta(\mathbf{x}, \mathbf{y}) = \arccos\left(\frac{\mathbf{x}^T\mathbf{y}}{\|\mathbf{x}\|\,\|\mathbf{y}\|}\right) \tag{2.13}$$

For text matching, the attribute vectors $\mathbf{x}$ and $\mathbf{y}$ are usually the term frequency-inverse document frequency (tf-idf) vectors of the documents.

Since the angle, $\theta$, is in the range of $[0, \pi]$, the resulting similarity will yield the value of $\pi$ as meaning exactly opposite, $\frac{\pi}{2}$ meaning orthogonal, 0 meaning exactly the same, with in-between values indicating intermediate similarities or dissimilarities.

## *2.4 Fusion*

In Air Force Doctrine, the Air Force warns its members not to strike targets based on single source intelligence; at some level, intelligence information should be fused together [64]. One form of this intelligence fusion is sensor fusion. Hall and Llinas state that multisensor data fusion combines data from multiple sensors to achieve improved accuracies and more specific inferences than could be achieved by using a single sensor alone [19]. The authors use the example of multisensory data fusion done by humans and animals to more accurately assess their surrounding environments. For example, the presence and quality of an edible substance may not be detected solely by sight, but by a combination of sight, touch, smell and taste [19].

Roli [55] explains that the growing interest in multiple classifier systems (MCS) is that, in many cases, the search for the best individual classifier for a specific application is either impossible or extremely difficult. Furthermore, complementary discriminatory information that multiple classifiers can exploit is lost by using a single classifier. Roli details two main phases of an MCS design: (1) the design of a classifier ensemble and (2) the design of a combination function. He also states that the literature typically focuses on only one of the two phases. Those methods that focus on classifier ensemble design strive to design an MCS made up of complementary classifiers that achieve optimal accuracy using a simple decision function. Those methods that focus on combination function assume that the individual classifiers that make up the MCS are all optimally designed within their own scope and, thus, optimality is obtained through formulating the optimal combination of those classifiers. These combination functions range from simple voting rules to "trainable" combination functions [55].

### *2.4.1 Constructing Classifier Ensembles*

Kittler *et al.* [29] also note that the design of a classifier ensemble is particularly useful when the individual classifiers are complementary. The authors state that classifier differences can be achieved through the use of different feature sets, different training sets, randomization or cluster analysis.

Input feature manipulation can be done manually or through an automated process so that individual classifiers are presented different feature sets. One method is to simply take a random subspace of features from the original feature space and then train a classifier on each subspace [4,23].

Output feature manipulation can be accomplished in the following manner [4]. For an $N$ class problem, partition the set of classes in such a way that each individual classifier solves a subset of the $N$ class problem. A combination method is then created so that the results of the subproblems solved by each individual classifier are combined to solve the original $N$ class problem.

Training set manipulation methods aim to achieve differences in classifiers by having $N$ different classifiers train on $N$ different training sets. Breiman [6] proposed a technique called bagging, which is an acronym for “bootstrap aggregating”. In bagging, multiple training sets are created by taking bootstrap samples of the original training data set. The bootstrapping procedure is as follows. For a training data set of size $n$, $X = (x_1, x_2, ...x_n)$, a new training data set of size $n$, $X^* = (x_1^*, x_2^*, ...x_n^*)$ is formed by taking a random sample of size $n$ from the original training data set, $X$, with replacement. Since each sample is drawn with replacement, each sample in $X$ can appear repeated times or not at all in $X^*$. This procedure can be repeated so that each of the $N$ individual classifiers has its own individual training data set.

One example of injecting randomness is random decision forests. In a decision forest classifier, a decision forest including multiple decision trees is used to classify "seen" training data and "unseen" data. Each individual tree performs an initial classification based on randomly selected subsets of the data. The classification outcomes by the individual trees are combined using a discriminant process in the decision-forest classier to render the ultimate classification decision [22].

### *2.4.2 Combination Functions*

Xu [75] explains that methods for fusing of multiple classifiers can be categorized by the type of information produced by the individual classifiers:

1. *Abstract-level outputs*: each classifier outputs a unique class label for each input pattern.
2. *Rank-level outputs*: each classifier outputs a list of possible classes, with ranking, for each input pattern.
3. *Measurement-level outputs*: each classifier outputs class confidence levels for each input pattern.

For abstract-level outputs, where each individual classifier outputs a unique class label for each input, a majority vote rule can be used to combine output labels [4, 28]. Consider outputs $S^{(1)}, ...S^{(N)}$ from $N$ abstract classifiers given an input pattern, $x$. The majority vote rule assigns a class label, $c_i$ to $x$ if $c_i$ is the most frequent label in the classifier outputs. Figure 2.3 depicts a majority voting rule for three classifiers. The majority voting rule is used primarily for an odd number of classifiers, thus avoiding ties.

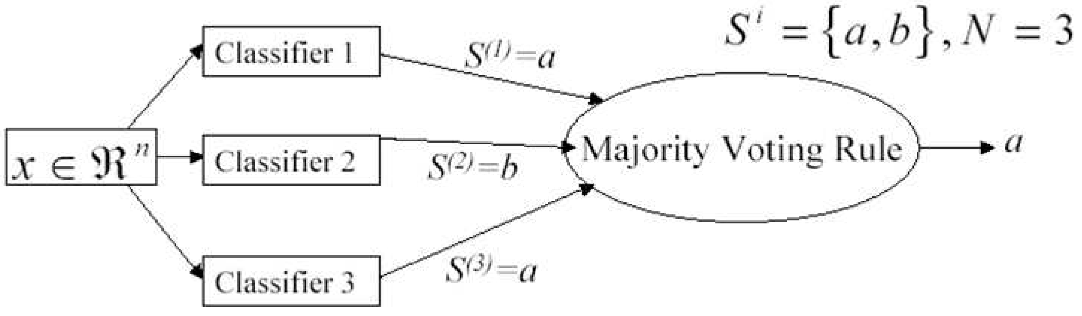

Figure 2.3 Majority Vote Fusion Fule [54].

Other abstract-level fusers, based on Bayes' formula, attempt to estimate the posterior probabilities of each input pattern through an independent training and validation set.

Rank-level fusion methods are used when the outputs of the individual classifiers are class scores or class probabilities. The rank-level fusers uses the ranked output from each of the individual classifiers and employs a certain methodology to determine the winning class. This type of scheme is well suited for cases when the winning class appears near the top of the list for each individual classifier, though not necessarily as the winning class. Figure 2.4 shows a rank-level fuser.

Measurement level classifiers take continuous outputs from the individual classifiers and combine them using some form of linear combination. The linear combiners can be a simple or weighted average. A simple average is optimal for classifiers

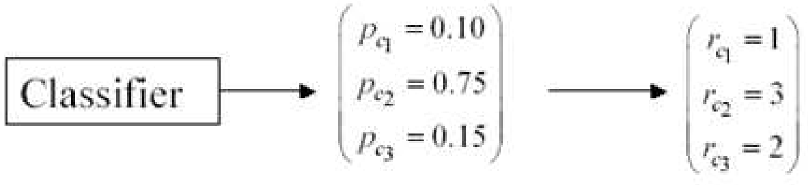

Figure 2.4 Rank Level Fusion Fule [54].

with the same accuracy, while a weighted average is indicated when the individual classifiers have unbalanced accuracies [4].

## *2.5 HRR Profiles Background*

The area of target recognition is a widespread research endeavour. From a military point of view, the central key to any successful air operation is the identification of targets. Air Force Doctrine Document (AFDD) 2-1 notes that if in order to utilize the unique range of air power, key targets must be identified [66]. With the advancement of technology and capabilities of sensors and processors, the ability to identify these key targets both accurately and timely seem within reach. Currently, the need for human analysts to examine data points to opportunities lost. Hebert [20] noted that the Unmanned Aerial Vehicle (UAV) Global Hawk is so effective that its full capability cannot be utilized. This is why automated analysis of sensor data is such a pressing matter. The target identification process can be broken down into two parts: detection and classification. We focus on classification is the subsequent section. First, we will consider the data sources from which a classification decision must be made. Next, we will examine some of the techniques in place for classification.

As defined in AFDD 2-5.2, the collective intelligence derived from visual photography, infrared sensors, lasers, electro-optics and radar sensors is collectively known as imagery intelligence [65]. Typical sensor types include electro-optical (EO), infrared (IR), synthetic aperture radar (SAR), high resolution range (HRR) radar and moving target identification (MTI) as well as multi spectral (MSI) and hyper-spectral imagery (HSI). The EO, IR and radar data provide single images, while the MSI and HSI provide multiple images of the same region from different frequency bands [3]. The focus of our research thus far has been in the application areas involving HRR data and HSI data, which we will discuss further.

## *2.6 HRR Processing*

In our main application, we use 2-dimensional X-Band SAR data on 15 separate targets collected by the Data Collection System (DCS) created by General Dynamics. These 15 targets along with their descriptions are shown in Table 2.2. This data was collected at both the HH and VV radar polarizations, which can lend itself to the treatment of two separate sensors for our experimentation purposes. A SAR system at the most basic level consists of a platform carrying a side looking antenna which illuminates an area of interest which electromagnetic radiation. Energy is reflected from objects, the magnitude of which depends on the composition of material, physical geometry, wavelength and polarization of the electromagnetic pulses as well as the azimuth angle. The belief in our research is that different objects should produce different radar returns, thus a unique HRR signature should exist for a unique target. We will discuss our ability to distinguish these signatures in the next section. For further discussion of properties or collection of SAR data, refer to Oliver [50].

Table 2.2 Targets Used for Collection with Descriptions and Characteristics.

| Type | Target Description | Tracks | Wheels | Gun |
|---|---|---|---|---|
| SCUD | Single Large Missile | N | 8 | N |
| SMERCH | MLRS Scud Confuser | N | 8 | N |
| SA-6 Radar | Soviet SAM Radar | Y | 0 | N |
| T-72 | Soviet Main Battle Tank | Y | 0 | Y |
| SA-6 TEL | 3 Medium SAMs | Y | 0 | N |
| Zil-131 | Medium Civilian Truck | N | 4 | N |
| HMMVV | Military SUV | N | 4 | N |
| M113 | Armored Personnel Carrier | Y | 0 | Y |
| Zil-131 | Small Civilian Truck | N | 4 | N |
| M-35 | Large Civilian Truck | N | 4 | N |
| SA-8 TZM | SA-8 Reload Vehicle | N | 6 | N |
| BMP-1 | Tank w/small turret | Y | 0 | Y |
| BTR-70 | 8-wheeled transport | N | 8 | N |
| SA-13 | Turret SAMs | Y | 0 | N |
| SA-8 TEL | Integrated Radar Exposed SAMs | N | 6 | N |

Several AFIT research efforts have focused on target classification using HRR signatures. MacDonald [42] applied Gaussian-mixture Hidden Markov Models to a three-class airborne target problem with Fourier transformed HRR signatures. This research found that forcing a relationship between the hidden states of the HMM and target orientation improved classification performance. DeWitt [10] processed HRR signatures produced by a synthetic CAD-based XPATCH model using the Prony technique in a two-class problem. The Prony technique generates feature vectors that describe scattering centers of the target. This research assumed prior target aspect angle knowledge within $\pm 5°$. Meyer's PhD research [44] considered invariant features drawn from sequenced HRR signatures and applied a template-based classifier for the resultant 3-dimensional scattering centers. This research showed that the effects due to white noise that degrade other classifiers are mitigated from the stability of the scattering centers. Using this approach, he was able to identify targets 20 percent obscured for up to 80 percent of his test cases.

Efforts from outside of AFIT include Williams *et al.* [71–73] who propose template-based ATR algorithms using HRR-derived features. They use a leave-one-out method to capture the effect of processing a SAR chip that is not in the template library for which the classifier trained. Under this method, a single target is left out of the training set, but all targets are used in testing. The process is then repeated for each target. Shaw *et al.* [59] conducted research using a template-based classifier with eigenvalues associated with HRR profiles across aspect angle. Zajik [76] employed wavelets-based features drawn from HRR profiles in a template-matching scheme. We detail some specific classifier types as applied to the HRR signature matching problem in the later sections.

## *2.7 Character Recognition Background*

Another application area we explore for our classification system is that of handwritten character recognition. Mantas [43] details the history of character recognition back to invention of the retinal scanner. Mantas' history continues with Nipkow's inventing of the sequential scanner up to David Shepard, founder of Intelligent Machine Research, Co. and the pioneer of commercial optical character recognition (OCR) equipment [43]. Mantas lists and defines four schemes of OCR:

1. *Fixed-font character recognition*: the recognition of specific fonts of typewritten characters.
2. *On-line character recognition*: the recognition of single hand-drawn characters where both the character image and time data are captured.
3. *Handwritten character recognition*: the recognition of single, hand-drawn characters, which are unconnected
4. *Script recognition*: the recognition of unrestricted handwritten characters, which may be connected and/or cursive.

Our specific focus will be on handwritten character recognition. Govindan [17] outlines three main ways character recognition methodologies can be looked upon. These three main ways are based on:

1. the approaches used,
2. the nature of applications and

3. the features used.

As our two classifiers are both feature based, we will focus further on this methodology. In terms of features used, Govindan [17] classifies character recognition techniques as either:

1. template matching and correlation techniques; or
2. feature analysis and matching techniques.

Template matching techniques compare an input character to a standard set of prototypes. The prototype that matches most closely to the input is the classification assignment for that input [17]. In feature analysis and matching, significant features are extracted from a character and compared to feature description of ideal characters. The ideal character whose description matches most closely provides recognition [17]. Though not stated explicitly, Govindan's explanation of feature analysis is nothing more than a comparison to a compressed prototype, such a mean. As we show in Chapter 3, the hybrid classifier we develop in this document can operate on any representation scheme or feature set. We thus have an ideal classifier to operate on the character recognition problem.

Previous work using template based character recognition was done by Connell [9]. This template matching technique considers the entire stroke of a character. Examples of these characters are shown in Figure 2.5. Strokes of a character are made up of a series of events, or feature vectors, which average 64 in number for all of the characters in their datasets. These events are represented by three measurements: the $x$ and $y$ offsets with respect to a reference coordinate and the angle of curvature of the written stroke at the sample point. The distance between any two aligned events is thus computed by considering the weighted differences between the three measurements. Strokes with different numbers of events are first aligned and differences between corresponding events are calculated, with total differences between strokes being the sum of the event differences along with stroke count difference penalty for strokes of different lengths. Prototypes are formed using clustering methods and classifications are made using nearest neighbor techniques and decision trees. Resulting digit recognition rates for a 10-class problem using this technique range from 86% to 91%.

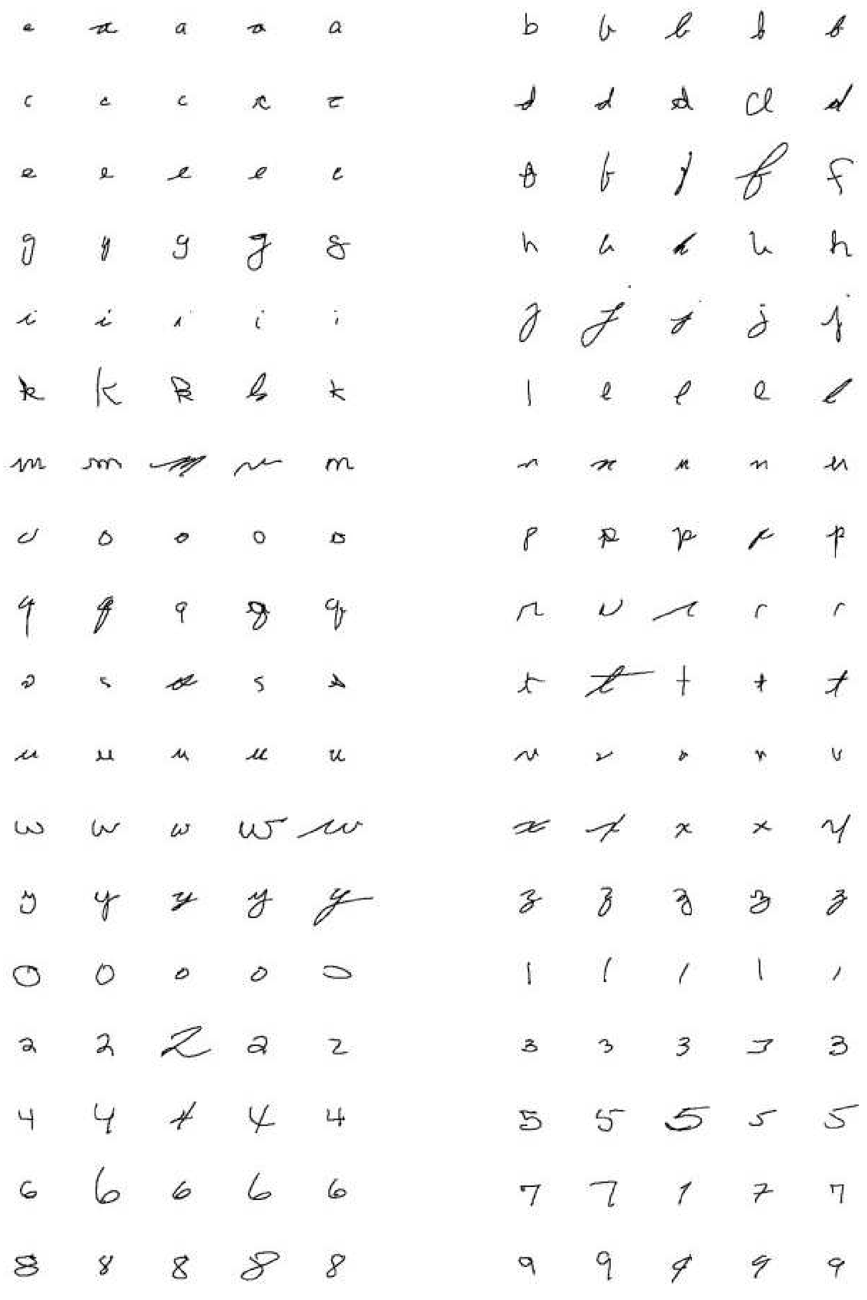

Figure 2.5 Examples of Characters from Connell [9].

# 3. Framework

We have already seen that there exist various approaches to pattern recognition problems [25]. We have also noted that almost every pattern recognition system falls into the category of a hybrid system. That is, most pattern recognition systems combine attributes of each of the general approaches, rather than being entirely a particular pattern recognition technique. These hybrid approaches combine preprocessing, segmentation, feature extraction, representation, classification, interpretation, etc. techniques from general classes of pattern recognition approaches to form a particular pattern recognition system.

The two pattern recognition systems we develop in this research use different representations, classification methods and non-declaration criteria. However, both systems build a template of prototype patterns to which new patterns are compared for the purpose of classification.

In this chapter, we begin with an overview of template-based classification systems. We begin with a discussion of the various components of a template classifier. This begins with the features extracted from each pattern and the representation scheme that is generated from those features. Next, we discuss various similarity metrics used in template classifiers. These similarity metrics are the basis for quantifying the level of likeness between two patterns which lead to classification decisions. Next, we present methodologies for both non-declaration and out-of-library decisions. Non-declarations occur when the classification system cannot distinguish the similarity a test pattern has to more than one of the classes in the template. Out-of-library determinations are made when the classification system indicates that a pattern is not among the types for which it has been trained to identify.

## *3.1 Template-Based Classification Overview*

Template classification is based on the notion of using characteristics or features of objects which may be used to train a classifier to recognize other objects with similar characteristics or features. In this approach, prototypes of all known patterns are stored into a template, which is used for comparison to unknown patterns for classification. The classification decision in template matching is based on a similarity measure. These measures can include correlation or Mahalanobis distance, as is used in Laine [34], Albrecht [3] and Friend [12]. In a forced decision scenario, the template whose similarity metric is smallest is deemed closest to the object to be classified and the corresponding class label is assigned to the object.

As discussed in Friend [12], SAR target signatures may vary significantly for a target of interest over small changes in aspect and depression angle as presented to a collection source. For this reason, it is common practice to collect many prototypes for the same target across different aspect and/or depression angles. Several factors contribute to the decision of how many templates should be collected. Among these include the amount of available information to generate templates, the amount of variance of the feature data and the storage and computation requirements needed for the features [12]. For this reason, the choice of features is of key importance in developing a template classifier.

### *3.1.1 Hybrid Pattern Recognition System*

Duda, Hart and Stork [11] detail how a pattern recognition system can be partitioned into distinct components. An example given in the text is shown in Figure 3.1. As detailed in the text, a sensor converts physical inputs such as images into signal data. A segmentor isolates image objects from background or other objects. A feature extractor measures properties from the objects that aid to categorize them as being of a certain object type, while distinguishing them from other object types. The classifier assigns objects to a category. The post-processor takes the output of the classifier and decides on recommended actions [11].

We describe our hybrid classification system in terms of these components. The first component is the sensing. The hybrid pattern recognition (HPR) system we develop begins by taking a sensor image and processing that image into a signal. This can be done by simply reshaping an $n$ by $m$ grayscale image into a 1 by $nm$ string or by using more complicated processing. In the case of the SAR imagery, this is done via algorithms obtained by AFIT from the authors of [73] associated with AFRL/SN. Once an image has been converted to a signal, the HPR system isolates the objects from background via a user defined noise threshold. This noise threshold works to filter the useful portion of the signal from the portion of the signal that can

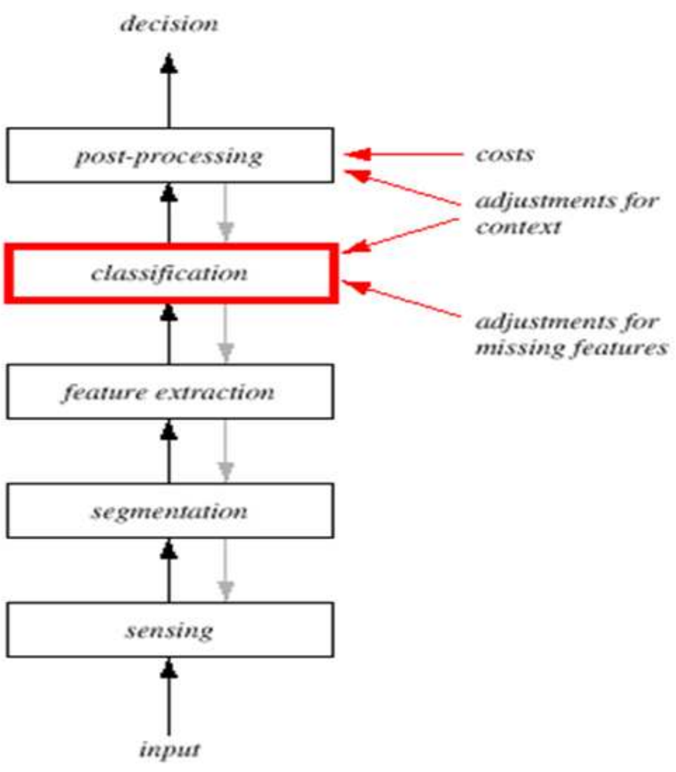

Figure 3.1 Notional Pattern Recognition System [11].

be construed as either noise or background clutter. This noise threshold can also be set to zero, if either a subject matter expert or the analyst concludes the entire signal is useful in making classification decisions. The top portion of Figure 3.2 shows an example of an HRR profile that was extracted from a SAR image via the AFRL algorithm. In the bottom portion, the HRR profile is run through a algorithm of the HPR system that quantizes the portion of the original signal that has not been filtered as noise. In this case, we represent noise using the integer 1 so that the graph of the representation is easier to see.

This noise filtering and quantization algorithm is the method of feature extraction used by the HPR. Duda, Hart and Stork [11] state that the goal of a feature

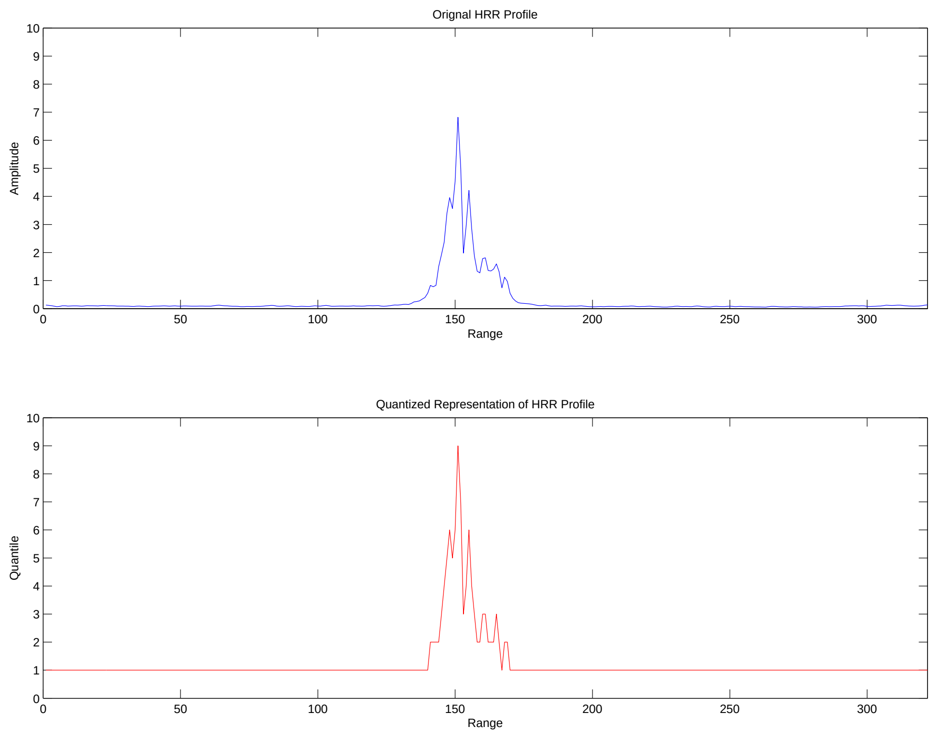

Figure 3.2  Example HRR Profile with Resulting Representation.

extractor is to find distinguishing features. That is, the feature extractor seeks to find measurements characteristic to a certain class of objects that are similar to all objects of that class and different from objects of other classes. As shown in Figure 3.2 the HPR system uses a representation scheme that is practical enough to accomplish classification, while still providing sufficient detail so that the original signal being represented can still be approximated from the representation.

The next component of the HPR system is classification. This consists of building a template of prototypes taken during the training phase. In the HRR application, these prototypes can be as precise as taken from each given aspect angle or can be made by combining various aspect angles into a mean prototype, as done in Koudelka *et al.* [32]. Other applications may only offer one prototype, or may suggest using a centroid of a set of prototypes, such as the mean of all prototypes for each class. The method of comparison between template and test exemplars is also made here. Examples of comparison metrics used in previous research include the Minkowski metric or the squared Mahalanobis distance. We detail each below.

For a point $\mathbf{x} = (x_1, x_2, ..., x_n)$ and a point $\mathbf{y} = (y_1, y_2, ..., y_n)$, the Minkowski metric of order $p$ is defined as:

$$d_p(\mathbf{x}, \mathbf{y}) = \left( \sum_{k=1}^{m} |x_k - y_k|^p \right)^{1/p} \tag{3.1}$$

$p$ need not be an integer, but it cannot be less than 1, because otherwise the triangle inequality does not hold.

The 2-distance is the Euclidean distance, $d_2$, a generalization of the Pythagorean theorem to more than two coordinates. It is what would be obtained if the distance between two points were measured with a ruler: the "intuitive" idea of distance.

The 1-distance is called the taxicab distance or Manhattan distance, $d_1$, because it is the distance a car would drive in a city laid out in square blocks (if there are no one-way streets).

Thus, two points are equal if and only if they are of the same length and each of their corresponding components are the same. The Minkowski metric is a generalization of more well known metrics such as the Manhattan distance, $p = 1$ or the Euclidean distance, $p = 2$.

Duda, Hart and Stork [11] explain Mahalanobis distance in the following manner. The multivariate normal density is completely specified by $d + \frac{d(d+1)}{2}$ parameters, namely the elements of the mean vector, $\mu$ and the independent elements of the covariance matrix, $\Sigma$. Samples drawn from a normal population tend to fall in a single cloud or cluster centered by the mean vector, $\mu$, with the shape of the cluster determined by the covariance matrix, $\Sigma$. The loci of points of constant density are hyper-ellipsoids for which the quadratic form

$$r^2 = (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)$$

is constant. The quantity $r^2$ is known as the squared Mahalanobis distance between $\mathbf{x}$ and $\mu$.

The final component of the HPR system is post-processing. This is where class membership decisions are made by using the above metrics or by extracting other measurements or quantities such as posterior probabilities from these metrics. In the case of classification decisions made from an edit metric, class membership would be assigned based on the smallest absolute value of edit metric between an exemplar to be classified from a given target template. In the case of a forced decision, this would be the final decision. We will discuss other methodologies such as out-of-library thresholds and non-declarations as aids to decrease misclassification in subsequent chapters.

### *3.1.2 Feature Vector Pattern Recognition System*

The feature vector pattern recognition system (FVPR) [3, 12, 34] we use is a slight modification from previous work. As previously applied to HRR profiles, the FVPR system preprocesses sensor images via the algorithms obtained by AFIT from the authors of [73] associated with AFRL/SN. Once a signal has been obtained from the image, a 10-dimensional feature vector is extracted by binning the HRR profile data into 10 equally sized range bins. The features are the maximum amplitude within each bin. This method of choosing peak amplitudes within range bins was based on research by Mitchell and Westerkamp [45] whose research introduced a statistical feature-based classifier which was applied to HRR signatures. This research, which is detailed in [46] used features extracted exclusively from the middle portion of the signal, where Mitchell states is the portion of the signal which contains useful information.

Once the feature vectors have been extracted from each HRR signal, templates are then formed for each target, where each template is made up of a feature vector at each aspect angle. In the classification phase, an unknown HRR profile is compared to the template in the following manner. Using a prior aspect angle knowledge of $\pm 22.5^\circ$, 360 template profiles are formed for each target using a total of 47 training profiles. These training profiles consist of the target template corresponding to the test template aspect angle, plus the 23 templates that precede the test template in aspect angle and the 23 templates that follow the test template in aspect angle. For instance, if a test profile has an assumed aspect angle of 45°, the template wedge used for comparison for each target would be made up from the template profiles having aspect angles from 22° to 68°. The mean vector, $\mu$ and covariance matrix, $\Sigma$ from these 47 template profiles are then used to compute the squared Mahalanobis distance between the test profile and the target template. This process is repeated for each target in the template library.

Once the squared Mahalanobis distances over all target templates have been computed, they are processed into a vector of posterior probabilities. Similar to the HPR system, in a forced decision scenario the target template having the minimum squared Mahalanobis distance to the test profile is deemed the winner and the test template is assigned the corresponding label.

## *3.2 Non-Declarations*

Friend [12] states that in all classification problems, decisions must be made that effect the overall quality of the classifier. Decision makers may impose constraints on a classifier due to their own willingness to risk critical and/or non-critical errors. This leads to the possibility of non-declarations. For a given exemplar, the choice to make a classification decision is usually based on thresholding the measurement the classifier employs to make decisions. Such measurements include, but are not limited to the output of Minkowski metric used by the hybrid classifier and the squared Mahalanobis distance used by the feature vector classifier. These thresholds create a rejection region which encompasses a certain interval of measures for which a classification decision is not made. The rejection region thus allows for a classification label when a classifier output falls outside the rejection region and disallows classification labels when a classifier output falls within the region. Previous non-declaration methods have included Chow [8], who stated that classification accuracy can be improved by withholding label assignments for exemplars which are difficult to classify. Chow's work used an optimal rule for rejection based on a single threshold for the posterior probability of a given class. For a classification problem with $N$ classes, withold making a classification decision for exemplar $x$ if the (winning) posterior probability for class $i$ given $x$, $P(\omega_i|x)$ is less than the threshold $T$. We can express this optimal rejection rule as, given $T \in [0, 1]$,

$$x \notin \omega_i \quad \text{if} \quad \max_{k \in 1,2,\ldots,N} P(\omega_k|x) = P(\omega_i|x) < T. \tag{3.2}$$

Fumuera *et al.* [15] showed that Chow's work could be improved by allowing for by-class thresholds, rather than a single threshold for all classes. The authors claimed this improvement by noting that if Chow's assumption of perfect knowledge of posterior probabilities were violated, no single threshold value could be used to find an optimal decision threshold. Using their by-class threshold scheme, a classification decision for exemplar $x$ is not made if

$$\max_{k\in 1,2,\ldots,N} \hat{P}(\omega_k|x) = \hat{P}(\omega_i|x) < \theta_i \qquad \text{where} \qquad \theta_i \in [0,1]. \tag{3.3}$$

where $\hat{P}(\omega_i|x)$ is the estimated posterior probability for class $i$ given $x$.

Laine [34] and Albrecht [3] implement a window-based non-declaration method. Their research used receiver operating characteristic (ROC) curve analysis as a method of establishing a non-declaration threshold. An example of a rejection region is shown in Figure 3.3.

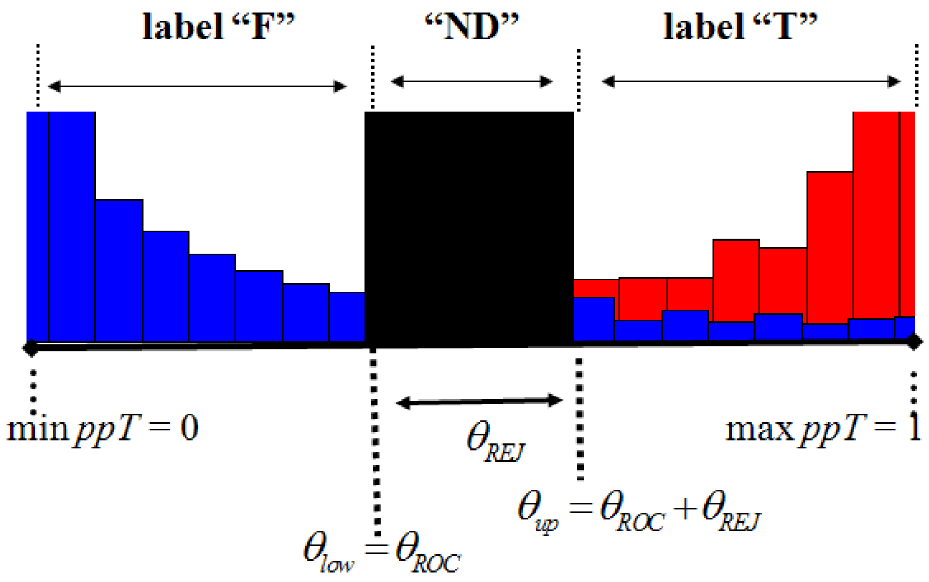
label "F"
"ND"
label "T"
$\min ppT = 0$
$\theta_{REJ}$
$\max ppT = 1$
$\theta_{up} = \theta_{ROC} + \theta_{REJ}$
$\theta_{low} = \theta_{ROC}$

Figure 3.3 Example Rejection Region for Two Class Problem [34].

A typical ROC curve shows tradeoff between true-positive probability and false-positive probability calculations by varying a threshold, $\theta \in [0,1]$, from 0 to 1.

A plot of the true-positive probability versus the false-positive probability at each $\theta$ generates the ROC curve. With the implementation of a rejection region, $\theta$ is used to define the region as follows. The center of the rejection region is defined to be $\theta_{ROC}$ and the half-width of the region is given by $\theta_{REJ} > 0$. Thus, as seen in Figure 3.3 the rejection region is defined by the interval $(\theta_{ROC} - \theta_{REJ}, \theta_{ROC} + \theta_{REJ})$. This rejection region serves as a means to overcome overlap between the two posterior probability distributions. Thus, the classifier will only assign a label where there is a high probability of class membership.

Friend [12] provided an alternative to this methodology by developing metrics based on entropy and Kullback-Liebler (KL) distance as non-declaration methods. In Friend's methodology, class sets in the feature space were targets. His entropy method treats target type as a random variable, $Y$ and treats each corresponding vector of posterior probability estimates as a probability mass function, $p_Y$, where $p_Y(y) = Pr\left\{Y = y\right\}, Y \in \mathbf{R}^n$. For a training set, the entropy,$H$ of each in library target's $10-$dimensional posterior probability measure is then calculated by

$$H(Y) = \sum_{i=1}^{10} p_y(y_i)\log_2\left(\frac{1}{p_y(y_i)}\right) = \sum_{i=1}^{10} -p_y(y_i)\log_2(p_Y(y_i)) = -E[\log_2(p(Y))].$$

Once the entropy has been calculated for each target in training set, a user defined quantile of the entropy scores is used to form the NDEC threshold that is used in testing. During the testing phase, each testing exemplar's entropy is calculated and compared to the NDEC threshold for the winning template. If the test exemplar's entropy exceeds the winning template class' NDEC threshold, the exemplar is assigned a NDEC label.

Friend's KL distance method again makes use of posterior probability estimates. His KL method treats target type as a random variable, $X$ and treats each corresponding vector of posterior probability estimates as a probability mass function, $p_X$, where $p_X(x) = Pr\{X = x\}, X \in \mathbf{R}^n$. Friend considers the true distribution function, $q_X$ as being 100% accurate, which has an entropy value of 0. As an example, he presents the true probability mass function for an exemplar, $x_i$ belonging to class i is

$$q_X(x_i) = \begin{cases} 1 & \text{if the exemplar belongs to class } i \\ 0 & \text{otherwise} \end{cases}$$

The KL distance between $q_X$ and $p_X$ is then

$$D(p_X, q_X) = \sum_{i=1}^{10} p_X(x_i) \log(\frac{p_X(x_i)}{q_X(x_i)}) = -\log(q_X(x_{i^*}))$$

where $i^*$ is the winning template. The implementation of the KL distance method follows the same steps as the entropy method for computing NDEC thresholds for each target template.

We extend the previous works by improving upon these methods. The idea of a single non-declaration threshold per class introduced in Fumera *et al.* [15] and implemented most recently by Friend [12] is extended by determining non-declaration status within the current exemplar itself. For our non-declaration scheme, a classification decision for exemplar **x** is not made if

$$\max_{k \in 1,2,\ldots,N} \hat{S_\alpha}(\omega_k|x) = \hat{S_\alpha}(\omega_i|x) < \theta_{i\alpha} \tag{3.4}$$

where $\hat{S_\alpha}(\omega_i|x)$ is the estimated similarity measure for class $i$ given $x$ at aspect angle $\alpha$.

Further, we note that difficulty in declarations is not simply due to the value of the winning score, but is largely due to the difficulty in distinguishing between at least two different classes. Thus, rather than simply thresholding on the single winning score, we will threshold on the difference between the class with the winning score and the class with the next closest score for exemplar $x$. Thus we have

$$\max_{k=1,2,\ldots,N} \hat{S_\alpha}(\omega_i|x) - \max_{k=1,2,\ldots,N, k\neq i} \hat{S_\alpha}(\omega_i|x) > \theta_{i\alpha} \tag{3.5}$$

where $\theta_{i\alpha}$ is some percentage of the overall range of scores for that exemplar.

We use the following non-declaration methodology. For a given exemplar, a non-declaration label (NDEC) is assigned to the vector **x** if the distance between the winning class' similarity metric and the next closest class' similarity metric is less than some percentage of the overall range of scores. As an example, consider the following vector of similarity metric scores:

$$S = (0.48, 0.42, 0.02, 0.02, 0.02, 0.02, 0.01, 0.01)$$

The winning score of $S_1 = 0.48$ and the next closest score of $S_2 = 0.42$ differ by 0.06. The entire range of these scores is $0.48 - 0.01 = 0.47$ If we use a non-declaration percentage is 10%, then we have $\theta_{i\alpha} = 0.47 * 0.1 = 0.047$. Since the difference of our largest two scores is larger than the 0.047, we would make a declaration, in this case class 1. On the other hand, if we use a non-declaration percentage of 20%, then $\theta_{i\alpha} = 0.47 * 0.2 = 0.094$, which is larger than our difference in scores and we would not make a declaration decision.

This new methodology improves Friend's methodology in the following ways. First, our methodology will make a non-declaration decision based upon how the exemplar compares to each of the in-library target templates. The threshold we compute will not be based on how any other exemplars may have compared to the template, which will hopefully give a more sound basis for whether the exemplar can distinguish which template it most resembles. Second, our methodology does not require the formulation of posterior probability estimates. Posterior probabilities have been characterized as troublesome in classification systems due to the need to estimate prior probabilities. Further, they are always normalized to sum to 1.0, which is not necessarily good under a forced decision [53, 56].

## *3.3 OOL Methodology Development*

In many classification problems, a complete set of possible objects is not known a priori. The use of NDEC labeling for objects where insufficient information exists to make a classification decision is one. With NDEC labeling, there is sufficient evidence to believe the object to be classified closely resembles at least one of the classes the classifier has been trained to recognize. The problem lies in the ability of the classifier to distinguish exactly which class the object most closely resembles. Thus, for example,the object in question belongs to either class $i$ or class $j$, but the classifier is unable to determine which.

Another growing trend in ATR classification is to provide a label for objects for which the classifier is not trained to recognize [3,12]. Such objects will be labeled out-of-library (OOL). Leap [36] describes the difficulty of OOL classes is from the fact that no means of training a classifier exists, since there are no exemplars of these class types. Leap expands the adage "We don't know what we don't know" with "but we do know that we don't know it". With this in mind, Leap deveoloped an OOL Detector that exploits the fact there are regions of the feature space in which no in-library classes are observed. Those regions serve as the points to be used as training points for out-of-library classes. Leap thus developed his OOL Detector by observing that if the feature space is chosen wisely, an attempt to distinguish regions within that feature space where our in-library class' features do not exist is possible.

This builds the hope for the development of an OOL detector, which we present in later sections.

To aid in distinguishing between different methodologies, we provide the following definitions for the different types of OOL methods [12].

1. *Transparent* Method based on complete knowledge of all test class types.
2. *Semi-Blind* Method based on partial knowledge about test class types. For example using some sort of descriptive statistics generated from test data to develop OOL thresholds.
3. *Blind* Method is based solely on in-library training data without any knowledge of test data class membership. OOL criteria is based entirely on in-library training data and the characteristics of a test exemplar.

Previous work with the classification of OOL objects has dealt with the formulation of an upper bound or threshold on training set measures as a means of determining OOL status. Williams *et al.* [71,73] discuss the use of object unknown to a trained classifier. In the case of their research, the authors use upper bounds on mean-square error as a threshold for identifying objects that are not contained in the training set. Ramamoorthy and Cassant [52] use MSTAR data with 8 in-library classes and 2 confusers with a feature space trajectory (FST) classifier. A test exemplar is rejected as OOL if the winning class' Euclidean distance exceeds an in-library threshold.

Albrecht [3] uses a heuristic which computes an OOL posterior probability, $pp_{OOL}$ for a test exemplar based on the in-library posterior probability estimate. This is accomplished as follows. For a test exemplar, a posterior probability vector, $x_{post}$ is computed. The posterior probability vector is then sorted in descending order. Thus, if for example the posterior probability vector is

$$x_{post} = [0.9, 0.01, 0, 0.05, 0, 0.02, 0, 0, 0.02, 0],$$

then the sorted posterior vector is

$$x_{sort} = [0.9, 0.05, 0.02, 0.02, 0.01, 0, 0, 0, 0, 0].$$

Albrecht then uses the assumption that a certain number of the values of the posterior vector are larger than the rest. In this example, only five of the 10 values in this posterior vector are nonzero. From that assumption, the threshold, $\theta^{(1)}_{OOL}$, to be used for OOL determination is taken by summing over the largest $n$ values in the posterior vector, where $n$ is a predetermined number determined in a training phase. In this case, if we assume $n$ to be 5, that is, $\theta^{(1)}_{OOL} = 5$ Albrecht's technique would sum over the 2nd thru the sixth posterior values to obtain $\theta^{(2)}_{OOL} = 0.1$. The value $pp_{OOL}$ is then based on the following heuristic:

$$pp_{OOL} = \begin{cases} 0 & \text{if } x_{OOL} < \theta^{(2)}_{OOL} \\ f(x_{OOL} - \theta^{(2)}_{OOL} & \text{if } x_{OOL} \geq \theta^{(2)}_{OOL} \end{cases}$$

where

$$\begin{aligned} \theta^{(2)}_{OOL} &= \text{Threshold obtained from a sub-optimization routine} \\ d &= x_{OOL} - \theta^{(2)}_{OOL} \\ f(d) &= \frac{2}{1 + e^{-10d}} \end{aligned}$$

If $x_{OOL} < \theta^{(2)}_{OOL}$, then the exemplar $\mathbf{x}$ is considered in-library, and the resulting probability of OOL is set to zero. Otherwise, if $x_{OOL} \geq \theta^{(2)}_{OOL}$, then the exemplar $\mathbf{x}$ is considered OOL and the probability of OOL is set to the value $f(d)$.

Friend [12] uses the following OOL methodology. His method begins by initiating a forced decision, he runs a training set of in-library targets through his feature-vector classifier. For each in-library target class, all correctly identified training exemplars are extracted along with their corresponding squared Mahalanobis distance, $r^2$. A threshold, $\theta_q$ is then generated using the $n$th quantile of the sorted $r^2$ values of each correctly identified exemplar per training class. In most cases, Friend states experimentation points toward using the quantile value of 1, meaning the OOL threshold for each target class will be the maximum observed $r^2$ of a correctly identified training exemplar for that target class. In testing, once the feature vector classifier identifies the winning template, the corresponding $r^2$ between the test exemplar and the winning template is compared to that winning template's threshold value. If the $r^2$ value exceeds the threshold, the exemplar is deemed to be OOL. Otherwise, the exemplar is assigned the label corresponding to the winning template.

Friend also discusses another method with a similar methodology. As with the OOL quantile method, a training set of in-library targets is run through his feature-vector classifier with a forced decision. For each in-library target class, all correctly identified training exemplars are extracted along with their corresponding squared Mahalanobis distance, $r^2$. A threshold for each target class, $\theta_m$ is then generated using the median, $\tilde{\theta_{tr}}$, and mean absolute deviation (MAD) over all values of $r^2$ for the correctly identified exemplars in each target class training set. Friend points out that the expectation is that $r^2$ values when comparing training exemplar to templates will be less than when making similar comparisons between test exemplars and templates. He therefore introduces the use of a constant multiple of the MAD, $\alpha$, which will be added to the median score of the $r^2$ values to form OOL threshold for each target class in the following manner. A test exemplar will be assigned an OOL label if the $r^2$ between the winning template and the test record exceeds the threshold $\theta_m$ corresponding to that target class, where $\theta_m = \tilde{\theta_{tr}} + \alpha MAD$. Friend did not pursue testing of this methodology due to the need to use OOL exemplars in the test set in order to find an optimal value for $\alpha$ for each target class. However, it should be pointed out that the quantile method as well as Albrecht's heuristic method would also require the use of OOL test data in order to optimize the associated parameter settings for each method. We therefore will develop an OOL method based on Friend's median/MAD methodology to use as a heuristic OOL methodology to use in conjunction with our non-numeric feature classifier.

The heuristic OOL methodology we develop follows the same basic steps as each of the methods done by Friend. Using a training set of in-library targets, we extract similarity metrics, $L$ for each correctly identified training exemplars. We then compute threshold values, $x_L$ in a similar fashion as the other heuristic methods. A threshold for each target class, $x_L$ is then generated using the mean, $\mu$ and standard deviation, $\sigma$ over all values of $L$ for the correctly identified exemplars in each target class training set. A test exemplar will be assigned an OOL label if the $L$ between the winning template and the test record exceeds the threshold $x_L$ corresponding to that target class, where $x_L = \mu + \sigma$. Like friend, rather than using a single standard deviation as the threshold distance, we can implement a multiple $\alpha$ so that our threshold value is computed as $x_L = \mu + \alpha\sigma$.

## *3.4 Fusion*

Our research examines the use of four different fusion schemes: (1) the Basic Ensemble Method (BEM), (2) Probabilistic Neural Network (PNN), (3) Borda Count and a (4) Bayesian Belief Network. Each of the four is discussed further below.

### *3.4.1 Basic Ensemble Method*

Perrone [51] proposes the use of averaged classifiers, such as the BEM. The authors prove that the mean square error is less for the ensemble than for the best classifier in the group. The presentation of the BEM combines a population of regression estimates to estimate a classifier function, $f(x)$ by using a set of functions $\mathcal{F} = \{f_1, f_2, ..., f_N\}$, such that each $f_i \in \mathcal{F}$ approximates $f$. The BEM regression function, $f_{BEM}$ is then defined as:

$$f_{BEM}(x) \equiv \frac{1}{N}\sum_{i=1}^{N} f_i(x) = f(x) - \frac{1}{N}\sum_{i=1}^{N} m_i(x) \tag{3.6}$$

where $m_i(x)$ is defined as the misfit of $f_i(x)$ from the true solution, that is, $m_i(x) = f(x) - f_i(x)$.

Perrrone and Cooper [51] proved that the mean square error for the BEM ensemble $MSE[f_{BEM}]$ is less than the mean square error for the best classifier in the group, $MSE[f_i]$ for $i$ total classifiers, that is,

$$MSE[f_{BEM}] \leq \min_{k \in 1,2,\ldots,i} MSE[f_k]$$

### *3.4.2 Probabilistic Neural Network*

Leap [35] describes the PNN fusion method as a simplistic fusion method that involves training a PNN on the posterior probabilities from the individual classifiers. The network in Figure 3.4 accomplishes classification for a two- class problem [68].

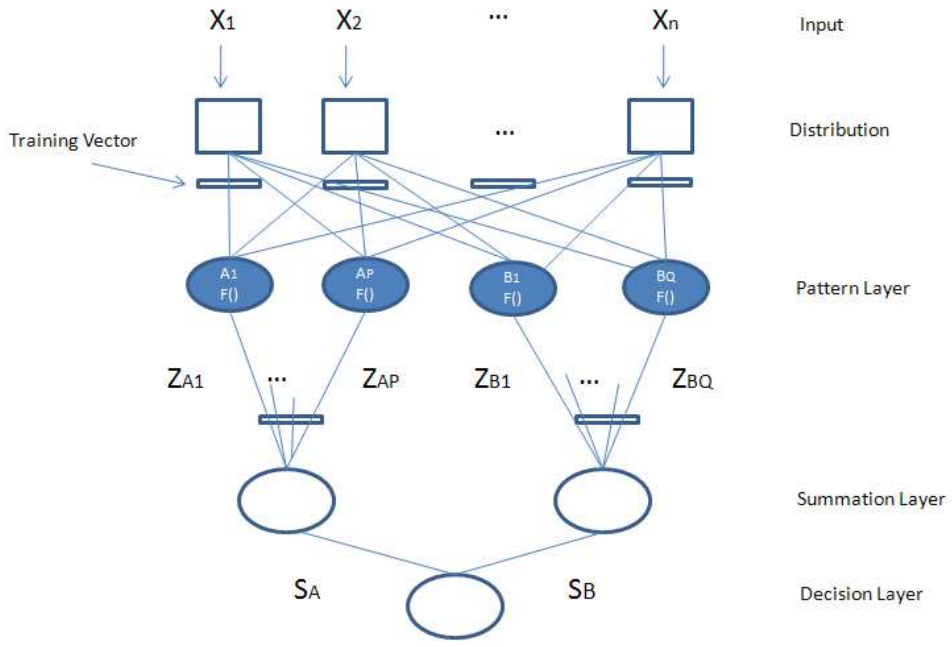

Figure 3.4 Probabilistic Neural Network [68].

Wasserman [68] describes the PNN as follows. This method is based on the assumption that the feature sets are normalized and independent and identically distributed multivariate normal with common variance, $\sigma^2$. The normalized input vector $X = (X_1, X_2, ..., X_n)$ is applied to the distribution layer neurons. This layer does not perform any computations, but merely serves as a connection point. Each training vector is used to calculate a set of weights, where each weight has the value of a component of that vector. Pattern layer neurons are grouped by the known classification of its associated training vector. Each pattern layer neuron sums the weighted inputs from every distribution layer neuron [68]. This is equivalent to taking the sum of squares of the training set and the test set, $(X - X_{R,i})^T(X - X_{R,i})$, where $X_{R,i}$ is the $i$th exemplar in the $R$th class from the training set. From normalization, this reduces to $(X_{R,i}^T X_i - 1)$. The pattern layer neurons then apply a non-linear function to the corresponding sum producing an output $Z_{ci}$, where $c$ indicates the true class of the training vector and $i$ indicates the pattern layer neuron. The nonlinear function for $Z_{c,i}$ is

$$Z_{c,i} = \exp\left(\frac{(X_{R,i}^T X_i - 1)}{\sigma^2}\right). \tag{3.7}$$

In this equation, $X$ is defined above and the set of weights corresponding to a pattern neuron represent a training vector $X_{R,i} = (X_{R,1}, X_{R,2}, , X_{R,n})$. The summation layer simply sums the $Z_{c,i}$ for each class [68]. Thus, the output of the summation layer for a specific class, $S_c$ is

$$S_c = \sum_{i=1}^{n} \exp\left(\frac{(X_{R,i}^T X_i - 1)}{\sigma^2}\right). \tag{3.8}$$

The decision layer compares $S_c$ for all classes and assigns the input vector to the class with the largest corresponding $S_c$.

### *3.4.3 Borda Count*

The Borda count fusion method is a group consensus function which maps a set of individual rankings to a combined ranking [23]. Using the posterior probability vector generated from a given classifier, a ranking for each class is generated, where the higher rank is associated with the class with the highest probability. This ranking continues sequentially for each of the possible classes. To make a classification decision, rankings are then combined additively. For any class $k$, the Borda count is the sum of the rankings from each individual classifier. If $B_i(k)$ is the ranking given to class $k$ by the $i$th classifier, then the Borda count for class $k$ is

$$B(k) = \sum_{i=1}^{n} B_i(k) \tag{3.9}$$

where $n$ is the number of classifiers [67].

The classification decision is based on the class with the highest overall magnitude. The Borda count method is simple to implement and requires no training. However, it does not take into account individual classifier capabilities.

### *3.4.4 Bayesian Belief Network*

The Bayesian belief network is a representation suited to the task of looking for relationships among a large number of variables [21]. A Bayesian network for a set of variables $\mathbf{X} = X_1, X_2, ..., X_n$ consists of a network structure, $S$ that encodes a set of conditional independence assertions about the variables in $\mathbf{X}$ and a set of probability distributions, $P$ associated with each variable. The network structure is a directed acyclic graph. The nodes in $S$ are in one-to-one correspondence with the variables in $\mathbf{X}$. Using $X_i$ to denote both a variable and its corresponding node, $\mathbf{P_i}$ to denote the parents of node $X_i$ in $S$, the joint probability distribution for $\mathbf{X}$ is given by

$$p(\mathbf{X}) = \prod_{i=1}^{n} p(X_i|P_i). \tag{3.10}$$

Because the Bayesian network for $\mathbf{X}$ determines a joint probability distribution for $\mathbf{X}$, the Bayesian network can be used to infer any probability of interest, such as the posterior probabilities for classification. Our research uses the open source Bayes Net Toolbox (BNT) for Matlab [1] for calculating posterior probabilities in our experiments.

## *3.5 Mathematical Framework*

### *3.5.1 Notation and Preliminaries*

We now develop a mathematical framework for our pattern recognition system. This framework, along with a mixed variable programming problem developed in the subsequent section enable us to optimize the parameters of our pattern recognition system in terms of some measure of performance, which we also present in a later section.

To begin, we consider a classification system, under a forced decision, that will be used to map a scene from a region of interest onto a label. For a two-class problem, let the label set be given by $\mathcal{L} = \{t_1, t_2\}$, where

$t_1$ denotes a target 1 label and

$t_2$ denotes a target 2 label.

Suppose we have identified representations $\mathbf{r}^{(1)}, \mathbf{r}^{(2)} \in \mathcal{R}$, where $\mathcal{R}$ is a set of possible representations, such that $\mathbf{r}^{(1)} \neq \mathbf{r}^{(2)}$, $\mathbf{r}^{(1)}$ corresponds to $t_1$ and $\mathbf{r}^{(2)}$ corresponds to $t_2$. Let $\mathbf{r} \in \mathcal{R}$ be a representation to which we wish to assign a label. Let $d \in \mathcal{D}$ be any metric defined on our set of representations, $\mathcal{R}$, where $\mathcal{D}$ is the set of all such metrics. Then we define the classifier $C_d : \mathcal{R} \rightarrow \mathcal{L}$ by

$$C_d(\mathbf{r}) = \begin{cases} t_1 & \text{if} \quad d(\mathbf{r}, \mathbf{r}^{(1)}) < d(\mathbf{r}, \mathbf{r}^{(2)}) \\ t_2 & \text{if} \quad d(\mathbf{r}, \mathbf{r}^{(2)}) < d(\mathbf{r}, \mathbf{r}^{(1)}). \end{cases}$$

This can be generalized for an $n$-class problem, with a label set $\mathcal{L} = \{t_1, t_2, \ldots, t_n,\}$, where $t_i$ denotes a label corresponding to target $i$ label for $i = 1, 2, \ldots n$.

We thus have $n$ representations $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \ldots \mathbf{r}^{(n)} \in \mathcal{R}$. If we now let $\mathbf{r} \in \mathcal{R}$ be a representation to which we wish to assign a label, our classifier $C_{d,} : \mathcal{R} \rightarrow \mathcal{L}$ is now defined by:

$$C_d(\mathbf{r}) = \begin{cases} t_i & \text{if} \quad d(\mathbf{r}, \mathbf{r}^{(i)}) < d(\mathbf{r}, \mathbf{r}^{(j)}) \quad \forall \quad j \neq i. \end{cases}$$

### *3.5.2 NDEC Model*

Now that we have a base model under a forced decision, we consider a classification system with a NDEC labeling option. For a two-class problem, let a label set be given by $\mathcal{L} = \{t_1, t_2, ndec\}$, where

$t_1$ denotes a target 1 label,

$t_2$ denotes a target 2 label and

*NDEC* denotes a non-declaration.

As in the forced decision case, we have representations $\mathbf{r}^{(1)}, \mathbf{r}^{(2)} \in \mathcal{R}$, where $\mathcal{R}$ is a set of possible representations, such that $\mathbf{r}^{(1)} \neq \mathbf{r}^{(2)}$, $\mathbf{r}^{(1)}$ corresponds to $t_1$ and $\mathbf{r}^{(2)}$ corresponds to $t_2$. Let $\mathbf{r} \in \mathcal{R}$ be a representation to which we wish to assign a label. Let $d \in \mathcal{D}$ be any metric defined on our set of representations, $\mathcal{R}$, where $\mathcal{D}$ is the set of all such metrics. Then we define the classifier $C_{d,\theta_1} : \mathcal{R} \to \mathcal{L}$ by

$$C_{d,\theta_1,}(\mathbf{r}) = \begin{cases} t_1 & \text{if} \quad d(\mathbf{r},\mathbf{r}^{(1)}) < d(\mathbf{r},\mathbf{r}^{(2)}) \ \wedge \ |d(\mathbf{r},\mathbf{r}^{(1)}) - d(\mathbf{r},\mathbf{r}^{(2)})| \geq \theta_1 \\ t_2 & \text{if} \quad d(\mathbf{r},\mathbf{r}^{(2)}) < d(\mathbf{r},\mathbf{r}^{(1)}) \ \wedge \ |d(\mathbf{r},\mathbf{r}^{(2)}) - d(\mathbf{r},\mathbf{r}^{(1)})| \geq \theta_1 \\ ndec & \text{otherwise.} \end{cases}$$

For a non-declaration threshold value $\theta_1 \in \Theta$. The development of the threshold $\theta_1$ is as follows. Consider the exemplar, $x_i$, with representation $r_i$. We wish to assign a label to this exemplar, thus we compute the distance between the exemplar and a prototype of each in-library class that is contained in a template of known classes. Let $\mathbf{r}^{(1)}$ denote the template prototype for class 1 and let $\mathbf{r}^{(2)}$ denote the class prototype for class 2. We compute the distance between $r_i$ and each class template. Using our choice of metric, $d \in \mathcal{D}$, we compute $s_1 = d(\mathbf{r}_i, \mathbf{r}^{(1)})$ and $s_2 = d(\mathbf{r}_i, \mathbf{r}^{(2)})$. Thus, we have a vector of distances between the exemplar and each target template, which we will call $S = (s_1, s_2)$. For this two-class problem we choose the winning class to be whichever class is the closest to the exemplar representation, thus we assign the exemplar to class $t_i$ if $d_i = \min(S) = \min(s_1, s_2)$. Because we now also include the NDEC labeling option, we set the following threshold. For the vector of distances, $S = (s_1, s_2)$, we set a non-delcartion threshold, $\theta = [max(S) - min(S)] * \gamma$, where $\gamma$ is a user defined multiple. Hence, for every exemplar, $x_i$, and for its corresponding representation, $r_i$, we define the non-declaration threshold for that exemplar, $\theta$, which is a function of the range of distances,$[max(S) - min(S)]$, and a user-defined multiple $\gamma$, where $0 \leq \gamma \leq 1$.

This can be generalized for an $n$ class problem, with a label set

$$\mathcal{L} = \{t_1, t_2, \ldots, t_n, ndec\},$$

where

$t_i$ denotes the label corresponding to target $i$ for $i = 1, 2, \ldots n$ and

$ndec$ denotes the label corresponding to a nondeclaration.

We thus have $n$ representations $\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \ldots \mathbf{r}^{(n)} \in \mathcal{R}$. If we now let $\mathbf{r} \in \mathcal{R}$ be a representation to which we wish to assign a label, our classifier $C_{d,\theta,} : \mathcal{R} \to \mathcal{L}$ is now defined by

$$C_{d,\theta}(\mathbf{r}) = \begin{cases} t_i & \text{if} \quad d(\mathbf{r}, \mathbf{r}^{(i)}) < d(\mathbf{r}, \mathbf{r}^{(j)}) \quad \forall j \neq i \\ & \wedge \quad |d(\mathbf{r}, \mathbf{r}^{(i)}) - d(\mathbf{r}, \mathbf{r}^{(j)})| > \theta_1 \quad \forall j \neq i \\ ndec & \text{otherwise.} \end{cases}$$

### *3.5.3 OOL Model*

Now that we have a NDEC model, we consider a classification system with a NDEC labeling option, that also includes OOL classes. With the introduction of OOL classes, we develop a method of identifying those classes seperate of the classifier. Our OOL Detector will take label and distance inputs from the classifier, and in turn will output a labeling decision of either IL or OOL. Thus, our classification system will become a composition of functions, where we compose the classifier and OOL Detector. We consider the two class problem first. For a two class problem, let a label set for the classifier be given by $\mathcal{L}^1 = \{t_1, t_2, NDEC, \}$, where

> $t_1$ denotes a target 1 label,
>
> $t_2$ denotes a target 2 label and
>
> $NDEC$ denotes a non-declaration.

Let a label set for the OOL Detector be given by $\mathcal{L}^2 = \{t_1, t_2, NDEC, OOL, \}$, where

> $t_1$ denotes a target 1 label,
>
> $t_2$ denotes a target 2 label,
>
> $NDEC$ denotes a non-declaration and
>
> $OOL$ denotes an out-of-library target.

As in the previous two cases, we have representations $\mathbf{r}^{(1)}, \mathbf{r}^{(2)} \in \mathcal{R}$, where $\mathcal{R}$ is a set of possible representations, such that $\mathbf{r}^{(1)} \neq \mathbf{r}^{(2)}$, $\mathbf{r}^{(1)}$ corresponds to $t_1$ and $\mathbf{r}^{(2)}$ corresponds to $t_2$. Let $\mathbf{r} \in \mathcal{R}$ be a representation to which we wish to assign a label. Let $d \in \mathcal{D}$ be any metric defined on our set of representations, $\mathcal{R}$, where $\mathcal{D}$ is a set of metrics. Then we define the classification system $C_{d,\theta} = C^2_{d_2,\theta_2} \circ C^1_{d_1,\theta_1} : \mathcal{R} \to \mathcal{L}$ by

$$C_{d,\theta} = C^2_{d,\theta_2} \circ C^1_{d,\theta_1}$$

where $C^1_{d,\theta_1} : \mathcal{R}^\backslash \rightarrow \mathcal{L}^1 \times \mathbb{R}^+$ is the classifier equipped with a NDEC option previously defined and $C^2_{d,\theta_2} : \mathcal{L}^1 \times \mathbb{R}^+ \rightarrow \mathcal{L}^2$ is an OOL detector. The classification system, equipped with an OOL detector, $C^2_{d,\theta_2}$ operates as follows. The classifier with a NDEC option, $C^1_{d,\theta_1}$ assigns an initial label, $l_i \in L^1$ where $s_i = min(S)$ is the metric output used for classification decsion, to an exemplar, $x$, where $S$ is the vector of outputs from the metric $d \in \mathcal{D}$ used by the classifier. Hence, $C^1_{s,\theta_1}(\mathbf{r}) = L_i$. This label assignment and metric output are fed to the OOL Detector, $C^2_{s,\theta_2}$, which uses the labeling assigment as an input and makes an in-library determination, based on some threshold $\theta_2$. First, if the exemplar is determined to be dissimilar enough to known in-lbrary targets, meaning $s_i > \theta_2$, then it is assigned on OOL label. In this case, we have $C^2_{d,\theta_2}(r_i) = OOL$. If the exemplar is determined to be in-library, meaning $d_i \leq \theta_2$, then we have $C^2_{d,\theta_2}(r_i) = l$. The exemplar is then further processed by the classifier, $C^1_{d,\theta_1}$ during which, NDEC determination is done and a final label assignment is made. Thus, our combined classification system, which combines the classifier and OOL detector has an over riding rule that makes a final label assignment. We can thus define the resultant classifier *system* $C_{d,\theta} = C^2_{d_2,\theta_2} \times C^1_{d_1,\theta_1} : \mathcal{R} \rightarrow \mathcal{L}^\in$ by

$$C_{d,\theta,}(\mathbf{r}) = \begin{cases} t_1 & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = t_1 \\ & \wedge \; C^2_{d,\theta_2}(\mathbf{r}) = t_1 \\ t_2 & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = t_2 \\ & \wedge \; C^2_{d,\theta_2}(\mathbf{r}) = t_2 \\ ool & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = t_1 \\ & \wedge \; C^2_{d,\theta_2}(\mathbf{r}) = OOL \\ \vee & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = t_2 \\ & \wedge \; C^2_{d,\theta_2}(\mathbf{r}) = OOL \\ ndec & \text{if } \; C^1_{d,\theta_1}(r) = NDEC \\ & \wedge \; C^2_{d,\theta_2}(\mathbf{r}) = t_1 \\ \vee & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = NDEC \\ & \wedge \; C^2_{d,\theta_2}(\mathbf{r}) = t_2 \end{cases}$$

An important note in the development of the composite classifier is the labeling hierarchy. If the OOL detector,$C^2_{d,\theta_2}$, outputs an $OOL$ label, this supercedes any other possible labeling assignment. On the other hand, if the OOL detector makes an $IL$ assigment, then the classifier $C^1_{d,\theta_1}$ must make an assignment from all other labels. This can be generalized for an $n$-class problem, with a label set $\mathcal{L} = \{\mathcal{L}^1, \mathcal{L}^2\}$. Where $\mathcal{L}^1 = \{t_1, t_2, \ldots, t_n, NDEC\}$ and $\mathcal{L}^2 = \{t_1, t_2, \ldots, t_n, NDEC, OOL\}$ as follows. The composite classifier $C_{d,\theta} = C^2_{d,\theta_2} \circ C^1_{d,\theta_1}$ assigns a label $OOL$ if $C^2_{d,\theta_2}(\mathbf{r}) = OOL$. Otherwise, the composite classifier assigns one of the remaining labels according to the output of $C^1_{d,\theta_1}$. Thus, for an $n$-class problem, we define the composite classifier *system* $C_{d,\theta} = C^2_{d_2,\theta_2} \circ C^1_{d_1,\theta_1} : \mathcal{R} \to \mathcal{L}$ by

$$C_{d,\theta,}(\mathbf{r}) = \begin{cases} t_i & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = L_i \quad \wedge \;\; C^2_{d,\theta_2}(\mathbf{r}) = IL \\ ool & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = L_i \quad \wedge \;\; C^2_{d,\theta_2}(\mathbf{r}) = OOL \\ ndec & \text{if } \; C^1_{d,\theta_1}(\mathbf{r}) = NDEC \;\; \wedge \;\; C^2_{d,\theta_2}(\mathbf{r}) = IL \end{cases}$$

We wish to develop an optimization framework for determining the best classifier performance. However, before we begin with the optimization framework, we must first develop the components of our classification system. We begin with the classifier itself. Now, for each possible choice of classifier, $C$, we have the following:

- $\theta \in \Theta \equiv$ the set of all thresholds associated with $C$
- $d \in \mathcal{D} \equiv$ the set of all possible metrics associated with $C$

The following diagram shows the components of a classification system

$$\mathcal{E} \overset{\mathcal{S}}{\to} \mathcal{D} \overset{P_{pre}}{\to} \mathcal{H} \overset{P_{rep}}{\to} \mathcal{R} \overset{C_{d,\theta}}{\to} \mathcal{L} \tag{3.11}$$

where:

- $\mathcal{E}$ is the population set of all scenes of interest;
- $\mathcal{S}$ is the set of all available sensors that map a scene onto an image;

- $\mathcal{D}$ is the set of all possible sensor images from sensor S;
- $\mathcal{P}_{pre}$ is the set of pre-processors that refine a image;
- $\mathcal{H}$ is the set of all refined images;
- $\mathcal{P}_{rep}$ is the set of all processors that map a refined image onto a representation;
- $\mathcal{R}$ is the set of all possible representations;
- $\mathcal{C}$ is the set of all available classifiers that map a representation onto a label;
- $\mathcal{L}$ is the set of all possible labels.

The composition of these mappings yield a classification system

$$A = C \circ P_{rep} \circ P_{pre} \circ S. \tag{3.12}$$

Moreover, since we are considering composite classification systems, we must consider fusion rules where

- $\mathcal{F}$ is a set of fusion rules.

For example, if we use fusion at the decision level for two classification systems, we will allow the two independent classification systems to make two independent decisions and use a fusion rule for making a final label decision. Classifier 1 will have its own set of parameters used to form a representation, which are generally different than the parameters used by classifier 2 in generating its representation. Once the two separate representations are formed, each of the classifiers will make its own label determination using its associated thresholds, parameters and edit metrics. In our example problems we fuse the hybrid classifier output with the feature vector classifier output, where the same sensor data is examined by each classifier. The two independent label outputs, $\mathcal{L}_1, \mathcal{L}_2$ are then fused via a single fusion rule, $f$, to form the final label output, $\mathcal{L}$. Figure 3.5 below depicts a classifier system that incorporates label fusion.

$$\begin{tikzcd}
 & & & \mathcal{R}_1 \arrow[r, "\mathbf{C_1}"] & \mathcal{L}_1 \arrow[dd, dash] & \\
\mathcal{E} \arrow[r, "\mathbf{S}"] & \mathcal{D} \arrow[r, "\mathbf{P_{pre}}"] & \mathcal{H} \arrow[ur, "\mathbf{P_{rep_1}}"] \arrow[dr, "\mathbf{P_{rep_2}}"'] & & \arrow[r, dotted, two heads, "f"] & \mathcal{L} \\
 & & & \mathcal{R}_2 \arrow[r, "\mathbf{C_2}"] & \mathcal{L}_2 &
\end{tikzcd}$$

Figure 3.5 Classifier System with Label Fusion.

We could instead incorporate fusion rules at any other level. For example, we could choose to fuse at the sensor data level, where we combine sensor images from two or more sensors operating on the same scene of interest. Suppose two separate sensors are operating in the same event space. Each sensor captures its own individual image. We could choose to fuse the sensor data prior to refining the image. Thus, we would take each sensor image and combine the data in some way to produce a third, combined image, which would then be processed as before. Figure 3.6 depicts a classifier system that incorporates data fusion.

$$\begin{tikzcd}
 & \mathcal{D}_1 \arrow[dd, no head] & & & & \\
\mathcal{E} \arrow[ur, "\mathbf{S}_1"] \arrow[dr, "\mathbf{S}_2"'] & {} \arrow[r, dotted, two heads, "f"] & \mathcal{D}_3 \arrow[r, "\mathbf{P_{pre}}"] & \mathcal{H} \arrow[r, "\mathbf{P_{rep}}"] & \mathcal{R} \arrow[r, "\mathbf{C}"] & \mathcal{L} \\
 & \mathcal{D}_2 & & & &
\end{tikzcd}$$

Figure 3.6 Classifier System with Data Fusion.

## *3.6 Mixed Variable Programming Formulation*

We must also consider a methodology for evaluating the performance of our classification system. Let $\rho$ be a real valued functional that quantifies the performance of our classification system, so that $\rho(A) \geq 0$, for each system $A$. We use the following mixed variable programming (MVP) formulation based on the previous work of Laine [34], Albrecht [3] and Friend [12]. For the MVP formulation, let $x$ be a vector of decision variables. Some of these decision variables are discrete, such as choice of fusion rule or sensor, while others are continuous, such as classifier thresholds or parameters [34]. We seek to find the optimal $x$ in the space of decision variables, $X$, given an objective function and constraints.

**Objective Function**

$$\max_{x \in X} \rho(x) \qquad \text{assuming maximization is optimal}$$

**Subject to:**

**Performance Constraints**

$$\begin{aligned}
E_{crit}(x) &< \Pi_1 && \text{upper bound on critical errors}\\
E_{ncrit}(x) &< \Pi_2 && \text{upper bound on non-critical errors}\\
P_{tp}(x) &> \Pi_3 && \text{lower bound on true positive rate}\\
P_{dec}(x) &> \Pi_4 && \text{lower bound on probability of declaration}\\
P_{lib}(x) &> \Pi_5 && \text{lower bound on out-of-library true positive rate}
\end{aligned}$$

**Sensor Selection Constraint**

$$\sum_{i=1}^{s} S_i \leq s \qquad \text{select from available sensors}$$

$$\sum_{i=1}^{s} S_i \geq 1 \qquad \text{select at least one sensor}$$

$$\text{where:} \quad S_i = \begin{cases} 1 & \text{if } i\text{th sensor is used} \\ 0 & \text{otherwise} \end{cases}$$

**Classifier Selection Constraint**

$$\sum_{j=1}^{c} C_k \leq c \qquad \text{select from available classifiers}$$

$$\sum_{j=1}^{c} C_k \geq 1 \qquad \text{select at least one classifier}$$

$$\text{where:} \quad C_j = \begin{cases} 1 & \text{if } j\text{th classifier is used} \\ 0 & \text{otherwise} \end{cases}$$

**Fusion Rule Constraint**

$$\sum_{k=1}^{f} F_k = 1 \qquad \text{select a single fusion rule}$$

$$\text{where:} \quad F_k = \begin{cases} 1 & \text{if } k\text{th fusion rule is used} \\ 0 & \text{otherwise} \end{cases}$$

**Minimum/Maximum Look Constraint**

$$\mathit{MinLook} \quad \leq NL \quad \leq \quad \mathit{MaxLooks}$$

where:

$$NL = \text{Number of looks at the target}$$

$$MinLook = \text{lower bound on number of target passes}$$

$$MaxLook = \text{upper bound on number of target passes}$$

**Budgetary Constraints** These constraints are subject to all budgetary limitations, which include but are not limited to: Research and Development, Operation and Maintenance, Procurement, Storage and Transportation.

$$\sum_{i=1}^{s} M_{S_i} S_i + \sum_{j=1}^{c} M_{C_j} C_j + \sum_{k=1}^{f} M_{F_k} F_k \qquad \text{limit costs of system}$$

where:

$M_{S_i} =$ the cost associated with sensor $i$

$M_{C_j} =$ the cost associated with classification system $j$

$M_{F_k} =$ the cost associated with fusion system $k$

**Physical System Constraints** These constraints are subject to the size limitations of the classification system as a whole. These constraints include, but are not limited to: weight, dimensions, bandwidth, platform, computation time and interface.

$$\sum_{i=1}^{s} P_{S_i} S_i + \sum_{j=1}^{c} P_{C_j} C_j + \sum_{k=1}^{f} P_{F_k} F_k \qquad \text{limit size of system}$$

where:

$P_{S_i} =$ the physical limit associated with sensor $i$

$P_{C_j} =$ the physical limit associated with classification system $j$

$P_{F_k} =$ the physical limit associated with fusion system $k$

**Threshold Constraints** These thresholds will differ in values and quantity for different sensors, fusion rules and classifiers. In the next section we discuss thresholds specific to the hybrid classifier.

$$\theta^{ijk} \geq \theta_{lower}$$

$$\theta^{ijk} \leq \theta_{upper}$$

where:

$\theta^{ijk}$ = a threshold associated with sensor $i$, classifier $j$ and fusion rule $k$

$\theta_{lower}$ = a lower bound for a given threshold choice

$\theta_{upper}$ = an upper bound for a given threshold choice

We can thus solve all, or more often times, a portion of this MVP formulation to achieve optimal parameter settings which are dictated by specific problem applications.

### *3.6.1 Threshold Considerations for the Hybrid Classifier*

Let us consider the specifics of threshold choices for the hybrid classifier we have developed. As we demonstrate in the formulation of the MVP, we have threshold choices that are specific to a given combination of sensor, fusion system and classifier. For the moment, we defer discussion of the first two components and focus on only the threshold decisions specific to our hybrid classifier. Table 3.1 lists the parameters associated with our classifier.

To consider further, let us examine these parameters in greater detail.

Table 3.1 Hybrid Classifier Parameters.

| **Parameter** | **Description** |
|---|---|
| $\theta_1$ | Noise Threshold |
| $Pre$ | Pre-processor choice |
| $d$ | Distance Measure |
| $W$ | Template Size |
| $T_R$ | Training Data Size |
| $Q$ | Number of Quantiles |
| $\theta_{OOL,i}$ | OOL Threshold for Class $i$ |
| $\theta_{NDEC,i}$ | NDEC Threshold for Class $i$ |

The choice of peak threshold, $\theta_1$, allows us to filter out apparent noise from a given signal. However, a choice of zero for this threshold will allow us to keep the entire signal, thus, extracting all information that may be contained in the signals. For example, previous work on HRR profiles [3, 12, 34] has filtered out what is assumed to be noise and only kept the middle part of the HRR profile for classification purposes.

The choice of preprocessing type, $Pre$, includes any and all steps done to the SAR image prior to forming representations. In our HRR profile case, the preprocessing steps include everything from the SAR chips processed using the AFRL algorithms, to include the interpolation done to mitigate the grouping of data.

The choice of a specific distance measure, $d$, is actually a method that defines the resultant classifier. Specific metrics, such as our absolute difference, the Manhattan distance, or the Euclidean distance, all contain properties specific to all metric spaces. Other choices of similarity quantifier, such as Mahalanobis distance or Z-score do not satisfy all of the properties of a metric space. These quantifiers do have other properties that allow one to assume statistical distributions of similarity and ease in computing posterior probabilities of class membership.

Template size, $W$ and training data size, $T_R$ are both problem specific. For example, in the presence of multiple exemplars of a given class, we can choose to use all exemplars of a certain class type to form a template in whole or by using a centroid such as the mean of the exemplars to be the prototype of that class. Training data size can vary due to availability of data as well as the need to learn parameters such as OOL or NDEC thresholds from training data.

In our classification scheme, we use quantization as a means to extract non-numeric features from patterns. Our choice for the number of quantiles, $Q$, contributes to both accuracy and computation time, thus the framework leads us to finding an optimal setting.

Finally, for a given class, $i$, the OOL threshold $\theta_{OOL,i}$ and the NDEC threshold, $\theta_{NDEC,i}$ are learned in the training phase. By learning how effectively the classifier identifies different classes, these thresholds are then formed to help optimize performance, within the framework.

Figure 3.7 shows a notional implementation of our methodology within the framework. Within this framework, two independent classifiers operate independently on sensor data. The sensors are independent in the sense that the same classifier is operating on two independent data sets, the HH polarized and the VV polarized data. For the purpose of experimentation, the two data sets are depicted to be from two independent sensors. The sensors each extract their own features and the corresponding classifiers make independent classification decisions. Each individual classifier passes information to the OOL detector, which makes decisions based on the information that is fed to it from each classifier. These classifiers then make their own classification decisions and use the mathematical framework to solve a sub-optimization problem, based solely on their own parameters and measures of performance. The individual optimal classifier outputs are then fused to produce the final label decisions. We will call this model the local optimization model, as it first optimizes the two sensor/classifier combinations prior to fusing the two classificatio system outputs together.

Figure 3.8 shows a notional implementation of our methodology, with a modification from the previously mentioned parallel sub-optimization technique. Here, each individual classifier goes through similar steps as in the previous technique. They each extract their own features, make class determinations and pass information to the OOL detector. However, rather than solving two sub-optimization problems and then using fusion to produce a final label, the fusion will be included in the optimization process. In this case, optimal parameter settings are found by finding the best overall fused result across all classifiers. We call this model the global optimizer, as it fuses all possible parameter combinations together prior to finding the optimum. We show the key change between the two models by highlighting the change in order of optimization and fusion between the local and the global models.

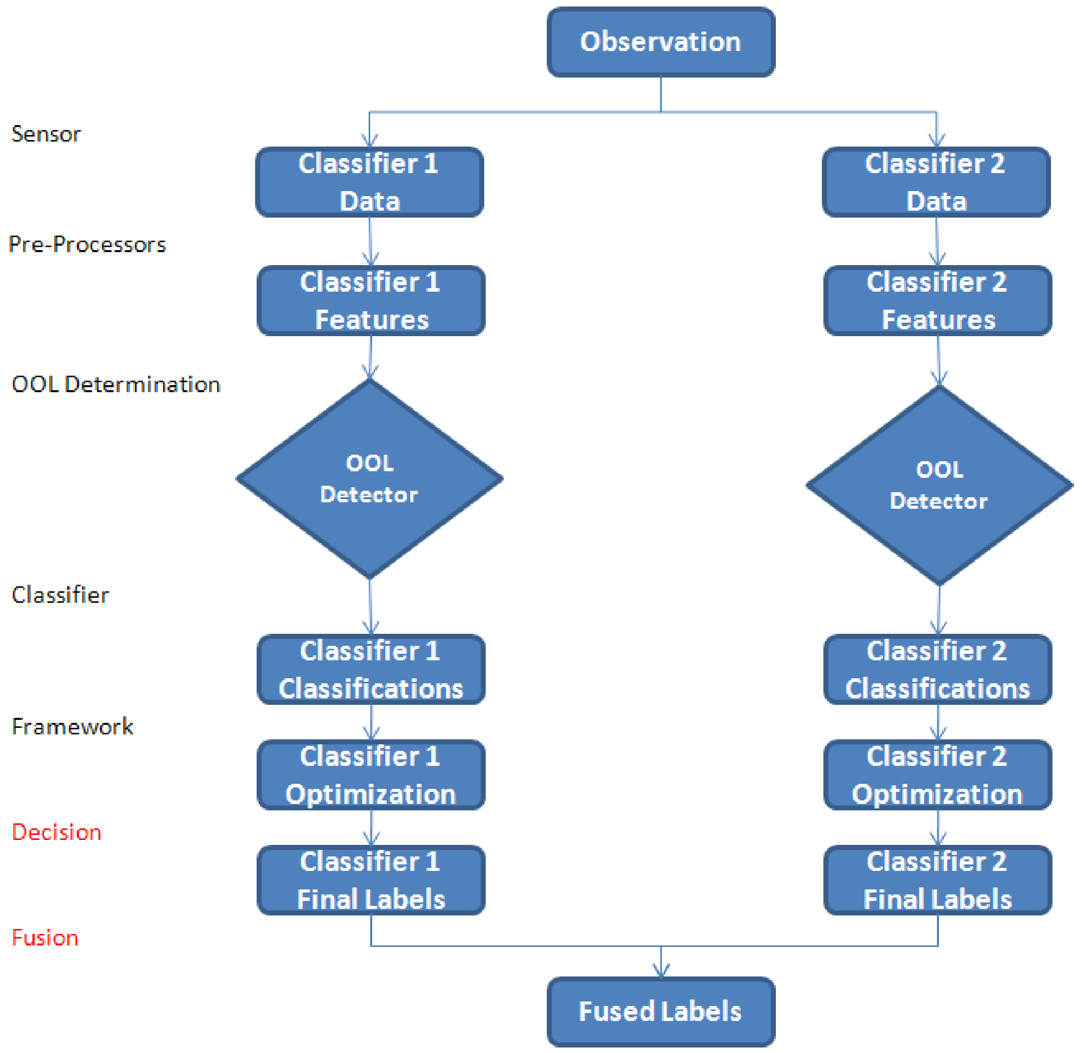

Figure 3.7 Classification System within mathematical optimization framework. In this process two independent classification systems operate on sensor data and produce their individual optimal outputs. These outputs are then fused to produce the final system output.

### *3.7 Methods of Evaluation*

In order to optimize the performance functional, $\rho$, of our pattern classification system, we must first determine how we will quantify performance. One method of presenting classification performance is through the use of a confusion matrix. A

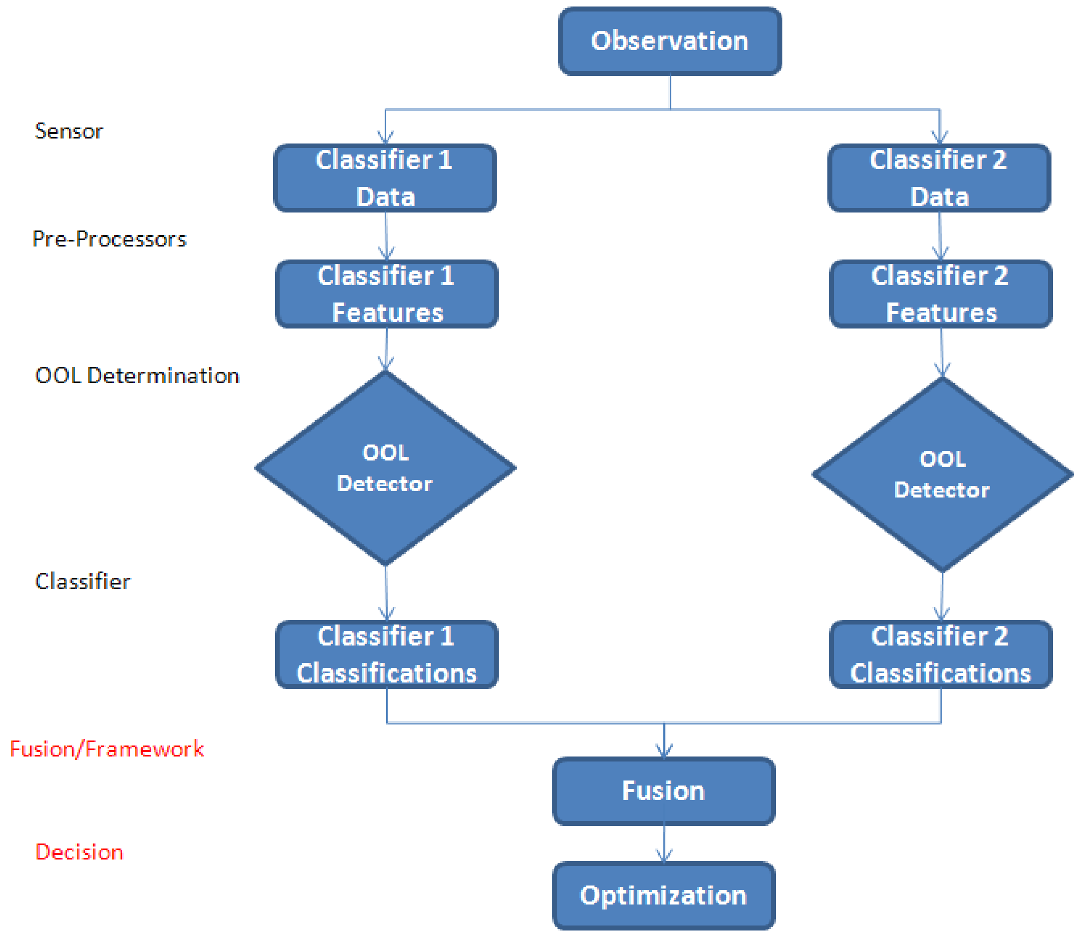

Figure 3.8 Classification System within mathematical optimization framework. In this process two independent classification systems are fused within the framework, thus producing the best overall parameter settings for the fused classification system.

confusion matrix for a classification problem considers the number of exemplars of a given class type and the number of labels of a given label type. So, as you add across the row of the confusion matrix, you will get the total number of exemplars of that class type. Similarly, as you add down the columns of the confusion matrix, you get the total number of labels used for the given label type. Figure 3.9 shows the entries of an example confusion matrix, as well as the row sums, $C_i$ and column sums, $L_j$. The classification problem represented by the confusion matrix has a total of $m$ classes and a total of $n$ possible labels. Note that these $m$ possible classes can include both in-library and out-of-library classes. Furthermore, the $n$ possible labels include both an out-of-library label as well as a non-declaration label.

$$\begin{array}{cc}
 & \text{Labels} \longrightarrow \qquad \text{Class Total} \\
\text{Classes} \downarrow & \left(\begin{array}{cccc} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{21} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{array}\right) \begin{array}{c} C_1 \\ C_2 \\ \vdots \\ C_m \end{array} \\
\text{Label Total} & \begin{array}{cccc} L_1 & L_2 & \cdots & L_n \end{array}
\end{array}$$

Figure 3.9 Confusion Matrix for $m$ classes and $n$ possible labels.

The first performance we consider is classification accuracy (CA). The CA for a given class is the number of successful times exemplars are correctly assigned to the class to which they belong. This is often referred to as the engineers' measure of effectiveness (MOE), since CA measures considers whether a classification system assigns a correct label, given a certain class. In practice, the classification accuracy for any given class, $i$ can be computed from the confusion matrix by dividing the $x_{ii}$ entry in the confusion matrix by its corresponding row sum. Thus, for the 10-class problem we have from finite data the estimate of CA for each class, $i$, denoted $\hat{CA}_i$

$$\hat{CA}_i = \frac{x_{ii}}{\sum_{j=1}^{10} x_{ij}}$$

Moreover, we can compute the average classification accuracy over all targets by taking the weighted average over all individual classification accuracies. If we assume equal prior probabilities, this is done by simply taking the mean of the ten individual classification accuracies. This computation is no different under the aggregated scenario, for which we have aggregated from ten classes to two classes.

Label accuracy (LA) measures the operational effectiveness of the classification system. This is often referred to as the warfighter's or user's MOE since this is an assessment of true class membership, given the classification system is indicating a certain label. Probabilistically, we have

$$p(C_i|L_j) = \frac{p(L_j|C_i)p(C_i)}{p(L_j)}$$

In practice, LA for any given class, $LA_j$ can be computed from the confusion matrix by dividing the $x_{jj}$ entry in the confusion matrix by its corresponding column sum. Thus, for the 10-class problem we have

$$\widehat{LA}_j = \frac{x_{jj}}{\sum_{i=1}^{10} x_{ij}}$$

The average LA over all ten targets can be taken as with the average CA, by taking the weighted sum over all ten labels. Furthermore, the aggregated LA for either of the two label types is done similarly for the aggregated CA case.

We thus define the following MOEs for a forced decision experiment as

$$\begin{aligned}
\widehat{CA}_{total} &= \frac{1}{10}\sum_{i=1}^{10} CA_i \quad \text{under the 10 true classes scenario} \\
\widehat{LA}_{total} &= \frac{\sum_{i=1}^{10} x_{ij}}{\sum_{j=1}^{10}\sum_{i=1}^{10} x_{ij} LA_j} \quad \text{under the 10 possible labels scenario} \\
\widehat{CA}_{H} &= \frac{\sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij}}{\sum_{i=1}^{5}\sum_{j=1}^{10} x_{ij}} \quad \text{under the aggregated 2 true classes scenario} \\
\widehat{LA}_{H} &= \frac{\sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij}}{\sum_{i=1}^{10}\sum_{j=1}^{5} x_{ij}} \quad \text{under the aggregated 2 possible labels scenario} \\
\widehat{CA}_{FN} &= \frac{\sum_{i=6}^{10}\sum_{j=6}^{10} x_{ij}}{\sum_{i=6}^{10}\sum_{j=1}^{10} x_{ij}} \quad \text{under the aggregated 2 true classes scenario} \\
\widehat{LA}_{FN} &= \frac{\sum_{i=6}^{10}\sum_{j=6}^{10} x_{ij}}{\sum_{i=1}^{10}\sum_{j=6}^{10} x_{ij}} \quad \text{under the aggregated 2 possible labels scenario}
\end{aligned}$$

The critical errors, $E_{crit}$ and non-critical errors $E_{ncrit}$ are computed as

$$
\begin{aligned}
E_{crit} &= \frac{\sum\limits_{i=6}^{10}\sum\limits_{j=1}^{5} x_{ij} + \sum\limits_{i=1}^{5}\sum\limits_{j=6}^{10} x_{ij}}{\sum\limits_{i=1}^{10}\sum\limits_{j=1}^{10} x_{ij}} \\
E_{ncrit} &= \frac{\sum\limits_{i=1}^{5}\sum\limits_{\substack{j=1\\ i\neq j}}^{5} x_{ij}}{\sum\limits_{i=1}^{5}\sum\limits_{j=1}^{5} x_{ij}} + \frac{\sum\limits_{i=6}^{10}\sum\limits_{\substack{j=6\\ i\neq j}}^{10} x_{ij}}{\sum\limits_{i=6}^{10}\sum\limits_{j=6}^{10} x_{ij}}
\end{aligned}
$$

The confusion matrices for the 10-class problem and an aggregated 2-class problem are shown in Figures 3.10 and 3.11. During our experiments, we aggregate from a 10-class problem to a 2-class problem by combining the first 5 target classes into one class and combining the second 5 target classes into a second class.

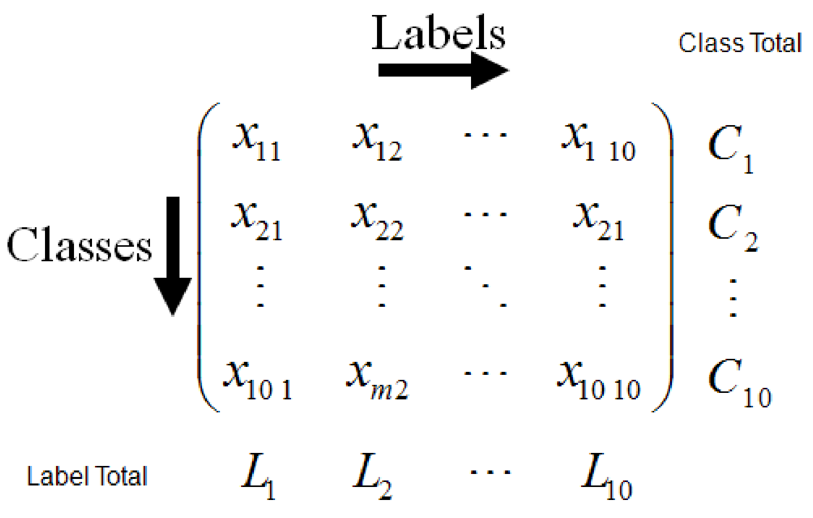
Labels
Class Total
Classes
$x_{11}$ $x_{12}$ $\cdots$ $x_{1\,10}$ $C_1$
$x_{21}$ $x_{22}$ $\cdots$ $x_{21}$ $C_2$
$x_{10\,1}$ $x_{m2}$ $\cdots$ $x_{10\,10}$ $C_{10}$
Label Total
$L_1$ $L_2$ $\cdots$ $L_{10}$

Figure 3.10 Confusion Matrix for 10 classes and 10 possible labels.

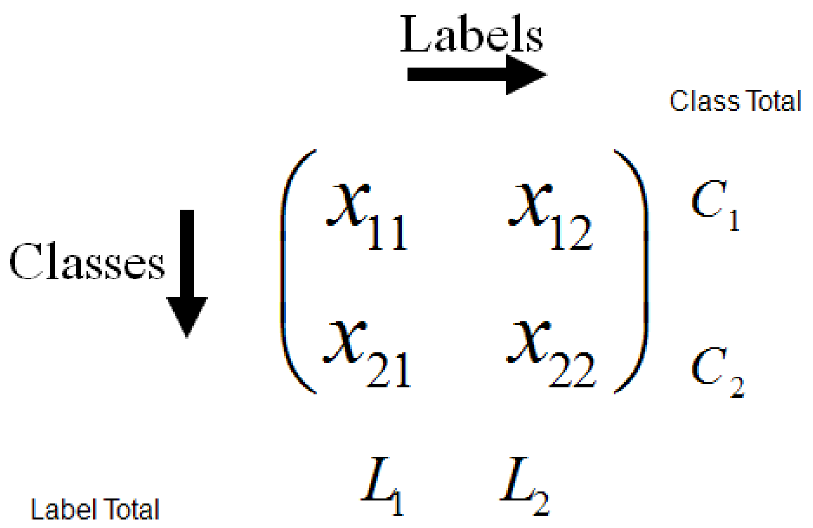

Figure 3.11 Aggregated Confusion Matrix for 2 classes and 2 possible labels.

# 4. Handwritten Character Recognition

## *4.1 NIST Data*

We now test our classification system methodology on the handwritten digit recognition problem. As stated by Liu [40], the performance of a classification system largely depends on the feature extraction approach and the classification/learning scheme. We examine four different feature extraction methods and report our findings on each. As has been done in previous testing methods, we use the Modified National Institute of Standards and Technology (MNIST) database of handwritten digits [37]. This database contains approximately 6000 training samples and approximately 1000 test samples of each digit (1-9). As described in Teow [60], the MNIST database was constructed from the National Institute of Standards and Technology (NIST) Special Database (SD) 3 and Special Database 1 which contain binary images of handwritten digits. NIST originally designated SD-3 as their training set and SD-1 as their test set. However, SD-3 is much cleaner and easier to recognize than SD-1. The reason for this can be found on the fact that SD-3 was collected among Census Bureau employees, while SD-1 was collected among high-school students. Drawing sensible conclusions from learning experiments requires that the result be independent of the choice of training set and test among the complete set of samples. Therefore it was necessary to build a new database by mixing NIST's datasets.

The original black and white (bi level) images from NIST were size normalized to fit in a 20 x 20 pixel box while preserving their aspect ratio. The resulting images contain grey levels as a result of the anti-aliasing technique used by the normalization algorithm. The images were centered in a 28 x 28 image by computing the center of mass of the pixels, and translating the image so as to position this point at the center of the 28 x 28 field.

The MNIST training set is composed of 30,000 patterns from SD-3 and 30,000 patterns from SD-1. The MNIST test set is composed of 5,000 patterns from SD-3 and 5,000 patterns from SD-1. The 60,000 pattern training set contains examples from approximately 250 writers. Figure 4.1 shows sample images from the MNIST database.

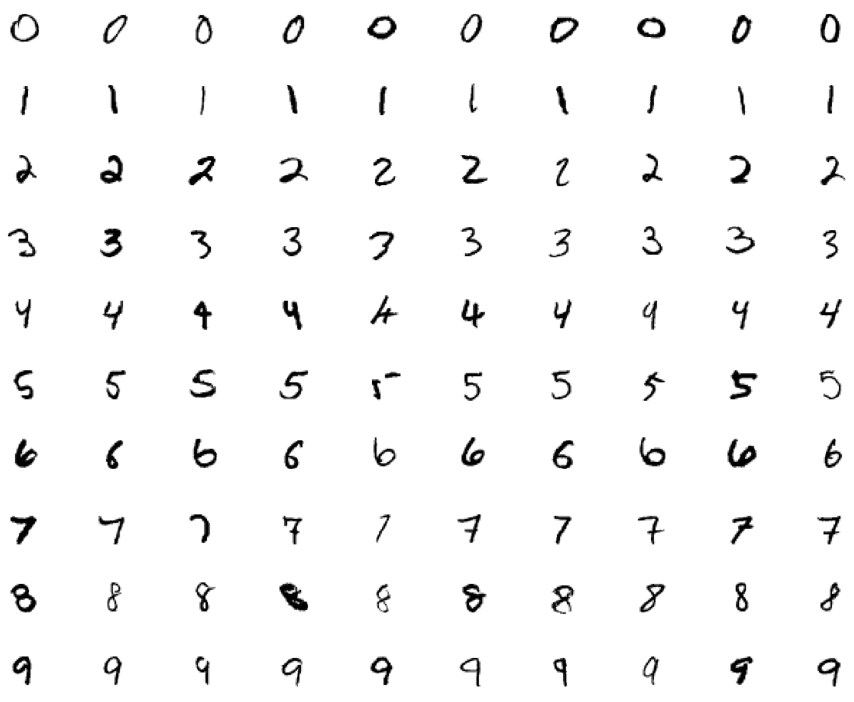

Figure 4.1 Sample Images From the MNIST Database.

Many previous methods have been tested on the MNIST database or on subsets of the database [40,60]. Thus, this database serves as an excellent baseline for making comparisons to other methods. Teow [60] tested a $k$-nearest neighbor classifier. The authors computed a distance (or similarity) between features of a test sample and the features of each of the training samples. Before the feature extraction step occurs each original 28 x 28 image is augmented by introducing a 2 pixel wide border around the image, resulting in a 36 x 36 pixel image. The features used in their methodology are extracted using a convolution mapping that detects neighboring pixels that have similar grey-scale values. The end result of their feature extraction is a feature vector of size 2592. The results reported used the same feature extraction method with two different quantifiers: (1) Euclidean distance and (2) cosine similarity measure. The experimental results reported showed that the best performing $k$ was for $k = 3$, where tests were performed for $k \in \{1, 3, 5, 7, 9\}$, but only the best results were presented. Those results are shown in Table 4.1. In both cases, the training set contained 60,000 samples and the test set contained 10,000 samples per class.

Table 4.1 Previous Results on MNIST Database [60].

| Similarity Measure | Error% | Reject% |
|---|---|---|
| Euclidean | 1.39 | 0 |
| Cosine | 1.09 | 0 |

Liu [40] reports 80 separate accuracies by taking combinations of eight different classifiers and 10 different feature vectors. Prior to feature extraction, each image is normalized to a standard size of 35 x 35. The features used in these experiments are different variants of a direction feature. The first variant is chaincode. In chaincode feature extraction, contour pixels from the normalized image are assigned 8 direction codes and the contour pixels of each direction are assigned to a direction plane. The next variant is gradient. In gradient feature extraction, different gradient operators were used to extract gradient strength and direction, which are transformed into feature vectors. The last variant used is peripheral direction contributivity (PDC). In PDC, the distance from a given stroke pixel to the nearest edge are computed in eight different directions. Feature measurements are then extracted from these directional components. The different classifiers used include a $k$-nearest neighbor classifier, three different neural network classifiers, a learning vector quantization classifier, a discriminative learning quadratic discriminant function classifier and two support vector machine classifiers. During experiments on the MNIST dataset, the feature vector scheme with the best average results was an eight direction gradient feature extractor of size 200. The best performing classifier was a support vector machine using a radial basis function (RBF) kernel. The authors note that this particular classifier has an extremely high storage and computation expense. The combination of this feature vector and classifier produced an accuracy of 99.227%, while the average CPU times in classifying a test pattern was 16.67 ms per test image. The authors conclude that this CPU time is prohibitive for real-time application of this classification methodology.

## *4.2 Implementation of the HPR System on the MNIST Data Set*

We now present our results from implementing the HPR classification system on the NIST data set. As described above, the NIST data set is the data set originally designated SD-3, from which the MNIST data set was constructed. Our experiments were run as follows. This database contains approximately 200 images per class. For our experiments, we randomly divide each class dataset into 150 training samples and 50 test samples. Features were collected by reshaping the original 16 x 16 pixel image into a 1 x 256 vector by either concatenating rows, concatenating columns or via a diagonalization. A representation is formed for each image from the training set and the test set using the quantization scheme described in the previous chapter. As a baseline of classifier performance prior to optimizing parameters, we use a noise threshold of zero and ten quantiles. Once the representations are formed, a prototype of each class is formed from the 150 training samples. In this case, we use the mean representation. Then, for each of the 50 test samples, the distance measure is computed between the test sample and each of the 10 prototypes. In this case, we use the Euclidean distance. The test sample is then classified according to the most similar prototype. The resulting confusion matrix, given in terms of percents, is shown in Table 4.2.

For this experiment, we observe an average classification accuracy, across all classes, of 77.63% and an average label accuracy of 83.27%. While these results are lower than previously reported results, the total CPU time for these tests was

Table 4.2 Confusion Matrix for the Recognition of NIST Handwritten Digits.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | **79.44** | 0.00 | 0.00 | 1.11 | 0.00 | 2.22 | 6.11 | 1.00 | 0.00 | 1.11 |
| 1 | 0.00 | **96.11** | 0.00 | 0.00 | 0.55 | 0.00 | 0.00 | 1.11 | 0.00 | 2.22 |
| 2 | 0.00 | 6.67 | **65.55** | 2.22 | 2.22 | 0.00 | 0.56 | 19.44 | 1.11 | 2.22 |
| 3 | 0.00 | 0.00 | 0.00 | **92.78** | 0.00 | 0.56 | 0.00 | 4.44 | 0.00 | 2.22 |
| 4 | 0.00 | 7.22 | 0.00 | 0.00 | **61.67** | 0.00 | 0.00 | 3.33 | 0.00 | 27.78 |
| 5 | 0.56 | 0.00 | 0.00 | 2.22 | 0.00 | **58.89** | 0.00 | 29.44 | 0.00 | 8.88 |
| 6 | 0.00 | 6.67 | 0.00 | 0.00 | 0.00 | 0.00 | **92.78** | 0.56 | 0.00 | 0.00 |
| 7 | 0.00 | 0.56 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | **94.44** | 0.00 | 5.00 |
| 8 | 0.00 | 9.44 | 0.00 | 6.67 | 0.00 | 0.00 | 0.00 | 18.33 | **19.44** | 46.11 |
| 9 | 0.00 | 0.56 | 0.00 | 0.00 | 0.56 | 0.00 | 0.00 | 2.22 | 0.00 | **96.67** |

virtually instantaneous, which indicates this methodology has potential for real-time applications.

We now apply the mathematical optimization framework to an experiment using the NIST database. We use our classification system within the framework across a variety of experimental parameter settings to find which parameter setting produce the best performance values. In this case, we use average classification accuracy ($CA_{total}$) and the average label accuracy ($LA_{total}$) to quantify performance. The parameters that will be varied in this experiment are the number of quantiles (NQ) used in representations and the distance measure(S) used. Here, we vary the number of quantiles used in the representation step from 5 to 15. The mathematical optimization formulation then becomes:

**Objective Performance Function**

$$\max_{x \in X} CA(x)$$

**Subject to:**

**Parameter Constraints**

$$\begin{aligned} NQ &\in \{5, 6, ..., 15\} \\ S &\in \mathbf{S} \equiv \{\text{Manhattan, Euclidean}\} \end{aligned}$$

We find that the Euclidean distance always outperforms the Manhattan distance in terms of our two measures of performance across all settings for NQ. Figures 4.2 and 4.3 show the average classification accuracy and average label accuracy across the number of quantiles used in representations, when using Euclidean distance. We find the best classification accuracy and the best label accuracy both occur when using seven quantiles.

### *4.3 NDEC Experimental Methodology*

For our initial non-declaration experiments, we will use the same 10 characters as in the previous experiments. Now, we will allow for non-declarations using the following method. For a given exemplar, $\mathbf{x}$, a non-declaration is made if the distance

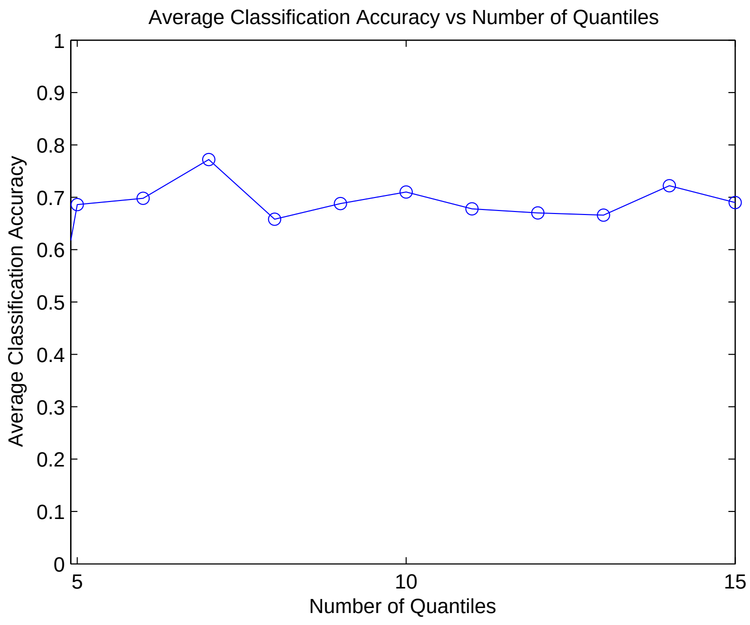

Figure 4.2 Average Classification Accuracy Across Number of Quantiles.

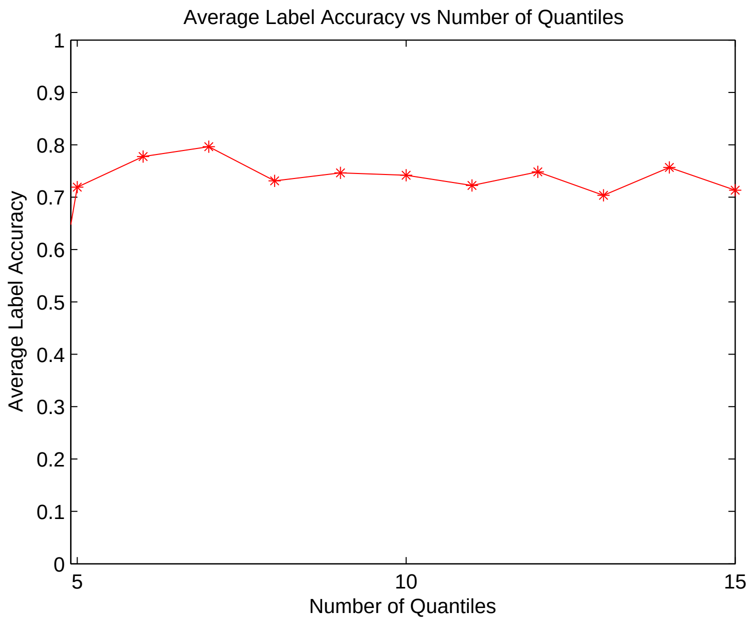

Figure 4.3 Average Label Accuracy Across Number of Quantiles.

between the exemplar and the winning template class $d(\mathbf{x}, \mathbf{x_1})$ and the next closest template class' distance , $d(\mathbf{x}, \mathbf{x_2})$, is less than some percentage of the overall range of scores.

We use our classification system within the mathematical optimization framework across a variety of experimental parameter settings to find which parameter settings produce the best performances. As in the forced decision scenario, we use total classification accuracy ($CA_{total}$) and the average label accuracy ($LA_{total}$) as the measures of performance. The parameters that will be varied in this experiment are the number of quantiles (NQ) used in representations, the non-declaration threshold $\theta_{NDEC}$ and the distance metric(M) used. Here, we vary the number of quantiles used in the representation step from 5 to 15, the non-declaration threshold from 0.00 to 0.05 and again use either the Manhattan distance or the Euclidean distance as our distance metric. The mathematical optimization formulation then becomes:

**Objective Function**

$$\max_{x \in X} CA(x)$$

**Subject to:**

**Parameter Constraints**

$$\begin{aligned} NQ &\in \{5, 6, ..., 15\} \\ \theta_{NDEC} &\in \{0.00, 0.01, ..., 0.05\} \\ S &\in \mathbf{S} \equiv \{\text{Manhattan, Euclidean Distance}\} \end{aligned}$$

The confusion matrix in Table 4.2 gives the optimal classification results for the above designed experiment. We present the average confusion matrix over 30 replications. The optimal parameter choices are given in Table 4.3.

Table 4.3  NIST Optimal Parameter Settings.

| **Parameter** | **Value Settings** |
|---|---|
| NQ | 7 |
| $\theta_{NDEC}$ | 0.01 |
| S | Euclidean Distance |

Using the optimal parameter settings of $NQ = 7$,$\theta_{NDEC} = 0.01$ and the Euclidean Distance for our choice of comparison metric we produce the confusion matrix in Table 4.4.

Table 4.4  Confusion Matrix for the Recognition of NIST Handwritten Digits with NDEC Threshold of 0.01.

| Class | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | NDEC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | **80.35** | 1.09 | 0.04 | 0.69 | 0.06 | 1.83 | 11.00 | 1.72 | 0.31 | 0.76 | 2.15 |
| 1 | 0.00 | **96.74** | 0.00 | 0.00 | 1.46 | 0.00 | 0.39 | 0.37 | 0.04 | 0.65 | 0.35 |
| 2 | 0.00 | 11.74 | **64.96** | 1.61 | 2.39 | 0.15 | 3.70 | 9.30 | 1.33 | 1.44 | 3.37 |
| 3 | 0.00 | 1.91 | 0.02 | **83.11** | 0.07 | 0.74 | 0.07 | 5.15 | 0.89 | 6.02 | 2.02 |
| 4 | 0.00 | 8.96 | 0.02 | 0.00 | **68.06** | 0.00 | 0.26 | 0.63 | 0.04 | 18.07 | 3.96 |
| 5 | 0.11 | 0.98 | 0.02 | 4.31 | 0.44 | **70.20** | 3.04 | 9.85 | 0.52 | 7.07 | 3.44 |
| 6 | 0.35 | 5.78 | 0.09 | 0.00 | 0.09 | 0.15 | **92.93** | 0.06 | 0.15 | 0.04 | 0.37 |
| 7 | 0.31 | 1.26 | 0.00 | 0.02 | 0.15 | 0.04 | 0.00 | **88.65** | 0.02 | 7.72 | 1.83 |
| 8 | 0.02 | 12.94 | 0.04 | 3.19 | 0.70 | 0.91 | 0.59 | 5.85 | **38.20** | 31.63 | 5.93 |
| 9 | 0.00 | 1.67 | 0.00 | 0.00 | 1.96 | 0.00 | 0.00 | 2.00 | 0.13 | **92.67** | 1.57 |

| | |
|---|---|
| Dec | 0.98 |
| AVG CA | 0.79 |
| AVG LA | 0.84 |

This error rate of 21%, while significantly greater than the previously reported results whose error rates were less than 1.5%, still demonstrates a favorable utility of tour classification. Moreover, the instantaneous CPU time indicates that further development of this classifier methodology on this application is feasible.

## *4.4 Summary*

The hybrid classifier applied to the MNIST data set yields several positive conclusions. First, the hybrid classifier was initially built with the intention of applying it to automatic target recognition problems. The fact that the hybrid classifier performs comparably to classifiers that were designed to be applied to optical character recognition applications demonstrates the versatility of the hybrid classifier. Second, the low storage and computation expense of applying the hybrid classifier on the optical character recognition problem indicates that the hybrid classifier could be modified and used in real time applications. Third, the utility of both the mathematical framework and the NDEC option within this application context show how we can boost classifier performance toward an optimum, without user interaction, thus indicating the usefullness of both methodologies.

# 5. Automatic Target Recognition

## *5.1 Overview*

The ability of a decision maker to make a quality real-time decision requires that reliable and timely information must be made available. Within the scope of combat identification, this means combat identification systems must be fast, accurate and easy to use. We now demonstrate how our classification system makes improvements to the timeliness and accuracy of combat identification systems by automatic target recognition methods. These improvements are the result of several key contributions. First, the development of a hybrid classifier that operates on a non-statistical set of features while maintaining comparable results to traditional statistical-feature classifiers gives us complementary classifiers whose outputs can be fused to improve overall system performance. The resulting classifier system, which we call a combined classification system, yields favorable results for both the individual classifier components as well as when fused together into the combined classifier.

The composite classifier is made up of different choices of pre-processor, out-of-library and non-declaration thresholds, different models or classifiers used, as well as choices of amount of signal or information from an observation that is retained. All of these decisions are managed through the mathematical framework which produces an optimal decision based on parameter choices. An overview of the combined classification system is shown in Figure 5.1.

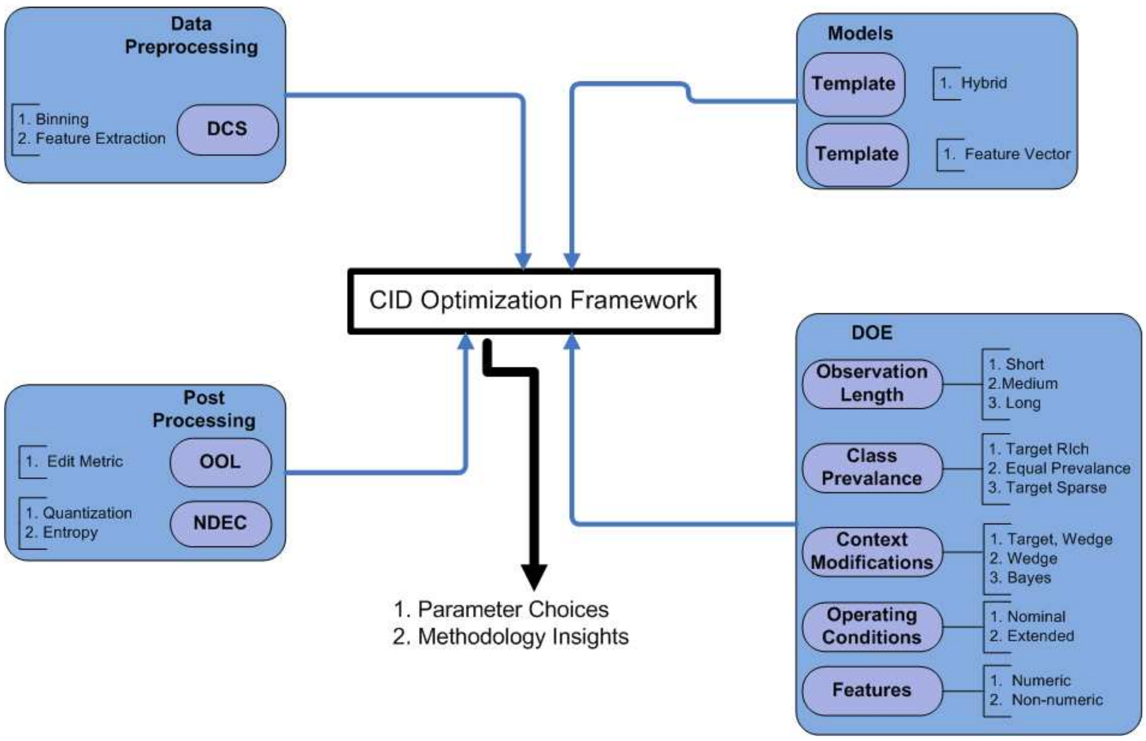

Figure 5.1 Overview of Combined Classification System.

## *5.2 HRR Data Description*

The high-range resolution (HRR) data set we use has been used in previous research at AFIT [3, 12, 34]. These dissertations detail the preprocessing methods that generate HRR profiles from SAR chips as well as the steps used to generate features. The original SAR data set was collected at Eglin AFB, FL from the General Dynamics Data Collection System (DCS). The SAR data was captured from two separate polarizations (HH and VV). These HRR profiles are then processed into representations or features that are used to make classification decisions. The two separate polarizations are treated as separate sensor data and are used for fusion experiments.

Each data set includes 724 SAR chips of each of the ten targets. A SAR system consists of a platform carrying a side-looking antenna, which illuminates a given area of interest with electromagnetic radiation pulses. As shown in Figure 5.2, the direction of the platform's travel is known as the azimuth and the perpendicular distance from the platform to the target is known as the ground distance, or range.

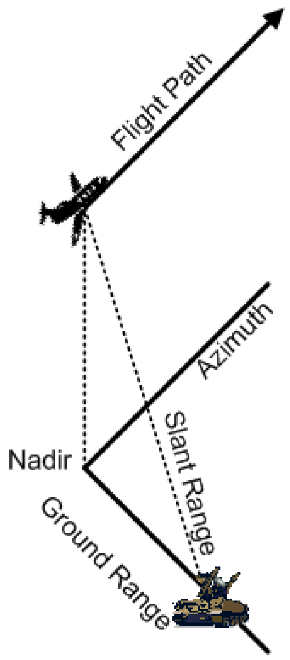

Figure 5.2 Diagram of SAR Collection Process [12].

During the collection process, each flight path encompassed approximately 90 degrees of aspect angle and successive spot collections were separated by approximately 4 degrees of aspect angle. In this collection, 32 passes were performed, providing 8 sets of SAR images collected over the entire 360 degrees of aspect angle. Figure 5.3 shows the aspect angle conventions for the data set.

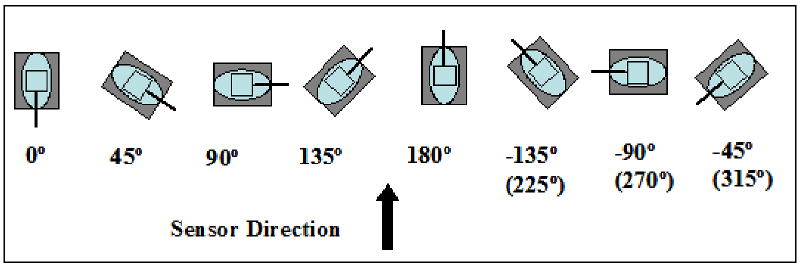

Figure 5.3 Aspect angle Conventions for DCS Data Set [12].

The data set of 724 exemplars for each data type were collected at a desired depression angle of between 6 and 8 degrees, though the actual range of depression angle was between 4 and 9 degrees. Figure 5.4 shows the relationship between platform and target in terms of depression angle.

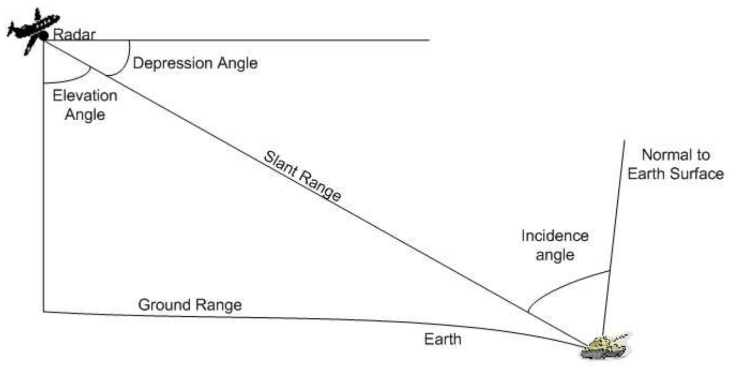

Figure 5.4 Diagram of Depression Angle [12].

Additionally, the data collection does not have an equal distribution of coverage over the 360 degrees of aspect angle. Friend [12] gave an example of this unequal distribution with a rose plot with 10 degree bins as in Figure 5.5. If there was an equal distribution across aspect angle, the bins of the rose plot would each contain 20 elements.

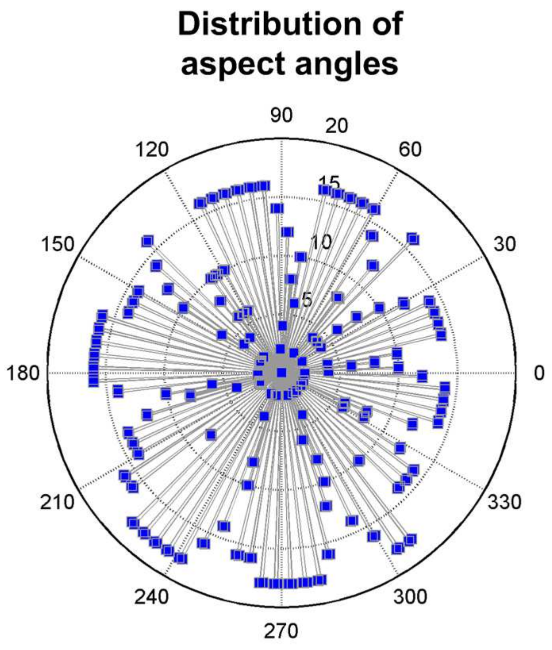

Figure 5.5 Example Aspect Angle Distribution. [12]

## *5.3 Implementing the Hybrid Classifier on the HRR Data Set with a Forced Decision*

The following is the methodology followed for classification experiments with the HRR profile data. The data set is composed of 724 exemplars of each target type. There are 10 in-library classes. The first five of these targets are hostile and the second five are non-hostiles. There are also 5 classes of out-of-library targets, which the classifier is not trained to recognize. Each exemplar has a range length of

322 and has a corresponding amplitude at each step that is plotted along the length to produce an HRR profile as shown in Figure 5.6.

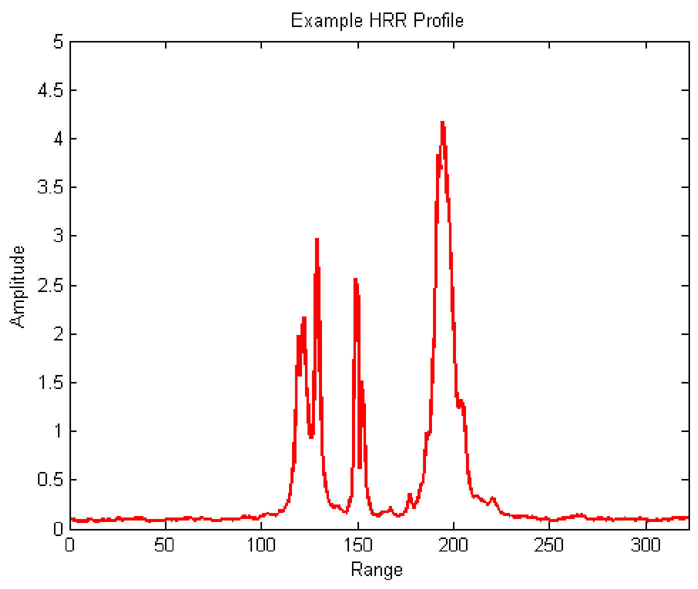

Figure 5.6 Example HRR Profile.

Each data set is first reduced in size to 723 by randomly removing a single exemplar. For each class, the remaining data is then randomly split into three separate data sets: a training set, a test set and a validation set. Each of these data sets contains 241 exemplars, which are linearly interpolated to form 360 exemplars in each set. These resulting exemplars each represent a single degree of aspect angle. The training set for each of the 10 in-library targets is used to form the templates in the two classifiers. The template for each class is formed in the following manner. Let $\alpha$ represent an aspect angle for an exemplar, where $\alpha \in \{1, 2, ..., 360\}$. Since aspect angle is not necessarily known with 100 % certainty, a symmetric wedge of 15 degrees is formed around the aspect angle estimate, $\alpha$ (i.e. $\alpha \pm 7$ degrees). For each of the 10 in-library targets, a template is formed for every aspect angle where $\widehat{x}_{\alpha d}$ represents the centroid estimate of the 15 degree symmetric wedge for the $d^{th}$ class and $\hat{\Sigma}_{ad}$ is a $10 \times 10$ estimated diagonal covariance matrix generated from the features in a 15 degree wedge centered at aspect angle $\alpha$ for the $d^{th}$ class. Once the template classifier is built using in-library targets, it is tested using two independent data sets containing both in-library and out-of-library targets.

As in previous work, [62,63] we assume a prior aspect angle knowledge of $\pm 7°$. Several works have proposed that effective pose estimation improves classification accuracy by reducing the number of templates that must be considered for classification decisions. Previous work with the feature vector classifier [3,12,34] follows the convention that pose estimation provided by a moving target indicator is accurate to within $\pm 22.5°$. As done in our previous work [62, 63] we use the more liberal estimate of $\pm 7°$ for both classifiers. Thus, when comparing an unknown exemplar with assumed aspect angle $\alpha$ to each of the in-library targets, we form a symmetric window of 15 degrees centered at $\alpha$ where $\widehat{x}_{\alpha d}$ represents the centroid estimate for the 15 degree window centered at $\alpha$ for the $d^{th}$ class.

Following the methodologies and notation used most recently by Friend [12], we adopt the following state space for classifier decisions. Consider the cases outlined in this research, where the general target classes will be: target of the day, other hostile target, or a non-hostile target. The state space for this scenario is outlined in Figure 5.7.

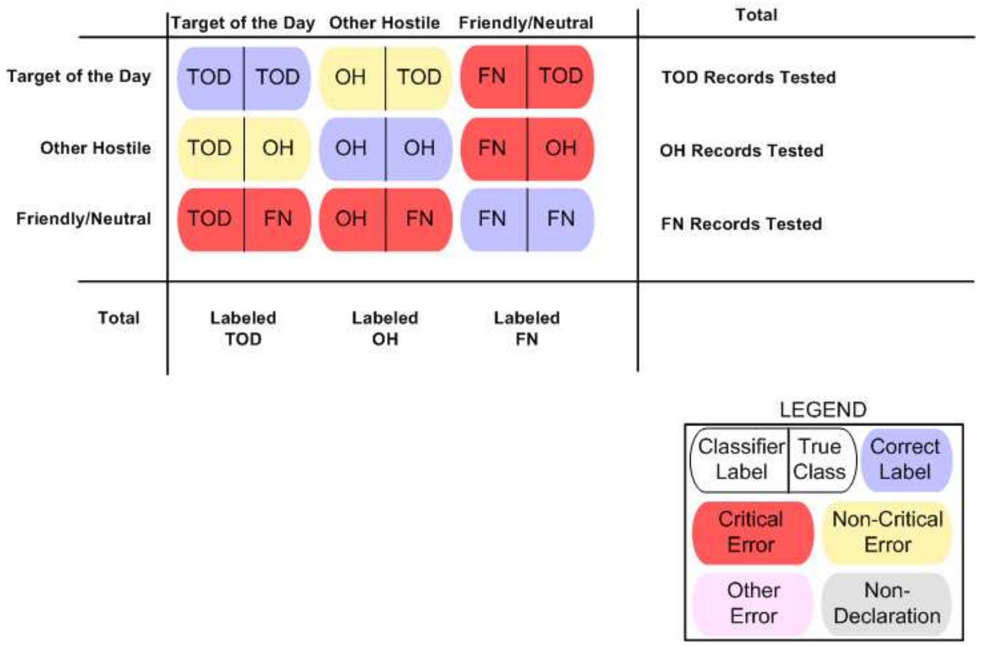

Figure 5.7 Event set for the forced decision classifier experiments [12].

A critical error, $E_{crit}$ is either the mislabeling of a hostile target as non-hostile or mislabeling of a non-hostile target as hostile. Either error presents the possibility of catastrophic consequences. A non-critical error, $E_{ncrit}$, would include mislabeling within aggregated hostile class or within the aggregated non-hostile class. While not necessarily catastrophic, these types of errors could lead to poor mission decisions, such as poor choice of weapon system or sortie generation.

Warfighter measures of effectiveness (MOEs) are used within the mathematical framework to evaluate system performance. In the following definitions, let $C_i$ indicate an exemplar truly belonging to a given target class, $i$. Let $L_j$ indicates an exemplar given a label assignment by the classification system that assigns the exemplar to target class, $j$. For a ten-class problem under a forced decision, Figure 5.8 shows the the resulting confusion matrix.

$$\begin{array}{cccccc}
 & \multicolumn{4}{c}{\text{Labels} \longrightarrow} & \text{Class Total} \\
\text{Classes} \downarrow & \left(\begin{matrix} x_{11} & x_{12} & \cdots & x_{1\,10} \\ x_{21} & x_{22} & \cdots & x_{21} \\ \vdots & \vdots & \ddots & \vdots \\ x_{10\,1} & x_{m2} & \cdots & x_{10\,10} \end{matrix}\right) & & & & \begin{matrix} C_1 \\ C_2 \\ \vdots \\ C_{10} \end{matrix} \\
\text{Label Total} & \begin{matrix} L_1 & L_2 & \cdots & L_{10} \end{matrix} & & & &
\end{array}$$

Figure 5.8 Confusion Matrix for the Forced Decision Classifier Experiments

Note that in this problem, the same numbers of both labels and classes exist. Thus, every exemplar is assigned to belong to one of the known classes. Furthermore, this 10-class confusion matrix can be aggregated to a higher level confusion matrix. Consider the confusion matrix in Figure 5.8. Now, we aggregate class membership as follows. We will consider two class types and similarly two label types. The first class type will be made up of the first five classes from our ten class scenario. The second type will be made up of the second five classes. Thus, we have aggregated from a 10-class problem to a 2-class problem. Under this aggregated scenario, we would consider a label to be correct, under the following. First, for any exemplar belonging to class $i \in \{1, 2, ..., 5\}$, a correct label would be an assignment of any label, $j \in \{1, 2, ..., 5\}$. Thus, an incorrect label for any exemplar belonging to class $i \in \{1, 2, ..., 5\}$ would be an assignment of label $j \in \{6, 7, ..., 10\}$.

The MOEs for this scenario include the following. First, classification accuracy (CA) is the rate at which exemplars are correctly assigned to the class to which they belong. This is often referred to as the engineers MOE, since CA measures the rate at which a classification system assigns a correct label, given a certain class type. In practice, the classification accuracy for any given class, $i$ can be computed from the confusion matrix by dividing the $x_{ii}$ entry in the confusion matrix by its corresponding row sum. Thus, for the 10-class problem we have

$$\widehat{CA}_i = \frac{x_{ii}}{\sum\limits_{j=1}^{10} x_{ij}}.$$

Moreover, we can compute the average classification accuracy over all targets by taking the weighted average over all individual classification accuracies. If we are using equal priors, this is done by simply taking the mean of the ten individual classification accuracies. This computation is no different under the aggregated scenario, except that we have aggregated from ten classes to two classes.

Label accuracy (LA) measures the operational effectiveness of the classification system. This is often referred to as the warfighter's or user's MOE since this is an assessment of true class membership, given the classification system is indicating a certain label. Probabilistically, we have

$$p(C_i|L_j) = \frac{p(L_j|C_i)p(C_i)}{p(L_j).}$$

In practice, LA for any given class, $LA_j$ can be computed from the confusion matrix by dividing the $x_{jj}$ entry in the confusion matrix by its corresponding column sum. Thus, for the ten class problem we have

$$\widehat{LA}_j = \frac{x_{jj}}{\sum\limits_{i=1}^{10} x_{ij}}$$

The average LA over all ten targets can be taken as with the average CA; by taking the weighted sum over all ten labels. Furthermore, the aggregated LA for either of the two label types is done similarly for the aggregated CA case.

$$\begin{aligned}
\widehat{CA}_{total} &= \frac{1}{10}\sum_{i=1}^{10} CA_i \quad \text{under the 10 true classes scenario} \\
\widehat{LA}_{total} &= \frac{\sum\limits_{i=1}^{10} x_{ij}}{\sum\limits_{j=1}^{10}\sum\limits_{i=1}^{10} x_{ij}LA_j} \quad \text{under the 10 possible labels scenario} \\
\widehat{CA}_H &= \frac{\sum\limits_{i=1}^{5}\sum\limits_{j=1}^{5} x_{ij}}{\sum\limits_{i=1}^{5}\sum\limits_{j=1}^{10} x_{ij}} \quad \text{under the aggregated 2 true classes scenario}
\end{aligned}$$

$$\widehat{LA}_H = \frac{\sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij}}{\sum_{i=1}^{10}\sum_{j=1}^{5} x_{ij}} \quad \text{under the aggregated 2 possible labels scenario}$$

$$\widehat{CA}_{FN} = \frac{\sum_{i=6}^{10}\sum_{j=6}^{10} x_{ij}}{\sum_{i=6}^{10}\sum_{j=1}^{10} x_{ij}} \quad \text{under the aggregated 2 true classes scenario}$$

$$\widehat{LA}_{FN} = \frac{\sum_{i=6}^{10}\sum_{j=6}^{10} x_{ij}}{\sum_{i=1}^{10}\sum_{j=6}^{10} x_{ij}} \quad \text{under the aggregated 2 possible labels scenario}$$

The critical errors, $E_{crit}$ and non-critical errors $E_{ncrit}$ are computed as

$$E_{crit} = \frac{\sum_{i=6}^{10}\sum_{j=1}^{5} x_{ij} + \sum_{i=1}^{5}\sum_{j=6}^{10} x_{ij}}{\sum_{i=1}^{10}\sum_{j=1}^{10} x_{ij}}$$

$$E_{ncrit} = \frac{\sum_{i=1}^{5}\sum_{\substack{j=1\\ i\neq j}}^{5} x_{ij}}{\sum_{i=1}^{5}\sum_{j=1}^{5} x_{ij}} + \frac{\sum_{i=6}^{10}\sum_{\substack{j=6\\ i\neq j}}^{10} x_{ij}}{\sum_{i=6}^{10}\sum_{j=6}^{10} x_{ij}}$$

The classification of ground vehicle targets from the DCS database is investigated using each of our classification schemes and the four fusion techniques. For our initial investigation, we consider the following. We first consider a forced decision scenario for ten targets from the DCS database. Each target type has two data sets, one from the HH-polarized data and the other from the VV-polarized data. We treat these data sets as sensor 1 and sensor 2, respectively. Table 5.1 details the ten targets used in the forced decision experiments with a description of each.

Table 5.1 Ten Class Forced Decision Targets.

| Tgt | Type | Description | Tracks | Wheels | Gun | Class |
|---|---|---|---|---|---|---|
| 1 | SCUD | Single Large Missile | N | 8 | N | Hostile |
| 2 | SMERCH | MLRS Scud Confuser | N | 8 | N | Hostile |
| 3 | SA-6 Radar | Soviet SAM Radar | Y | 0 | N | Hostile |
| 4 | T-72 | Soviet Main Battle Tank | Y | 0 | Y | Hostile |
| 5 | SA-6 TEL | 3 Medium SAMs | Y | 0 | N | Hostile |
| 6 | Zil-131 | Medium Civilian Truck | N | 4 | N | Friendly |
| 7 | HMMVV | Military SUV | N | 4 | N | Friendly |
| 8 | M113 | Armored Personnel Carrier | Y | 0 | Y | Friendly |
| 9 | Zil-131 | Small Civilian Truck | N | 4 | N | Friendly |
| 10 | M-35 | Large Civilian Truck | N | 4 | N | Friendly |

### *5.3.1 The Hybrid Classifier: Forced Decision*

We will now utilize the optimization framework for the forced decision scenario. We consider two separate classifiers both operating within the framework across various parameter settings. These parameter settings are shown in Table 5.2. For this scenario, the parameters being explored are: fusion method, $F$; wedge size, $W$; the number of quantiles, $Q$, used in forming representations; the noise threshold, $\theta_1$; and finally, the choice of comparison metric, $M$, where we will use the Minkowski metric for $p$ = 1,2 or 3. As previously defined, the Minkowski metric is the Manhattan distance when $p = 1$ and the Euclidean distance when $p = 2$.

Here, the hybrid classifier operates in one of the two sensor data sets, the HH or VV polarized data using the full HRR profile for each exemplar. Within the

Table 5.2 Hybrid Classifier Settings.

| **Parameter** | **Value Settings** |
|---|---|
| W | 7,15,22 |
| Q | 5,6,...,15 |
| $\theta_1$ | 0.0, 0.05, 1.0 |
| M | Minkowski for $p = 1, 2, 3$ |
| F | BEM,Borda Count, PNN, Bayes Net |

framework, each individual classifier/sensor combination is optimized. Model 1 is the local optimimization model, which is shown in Figure 5.9.

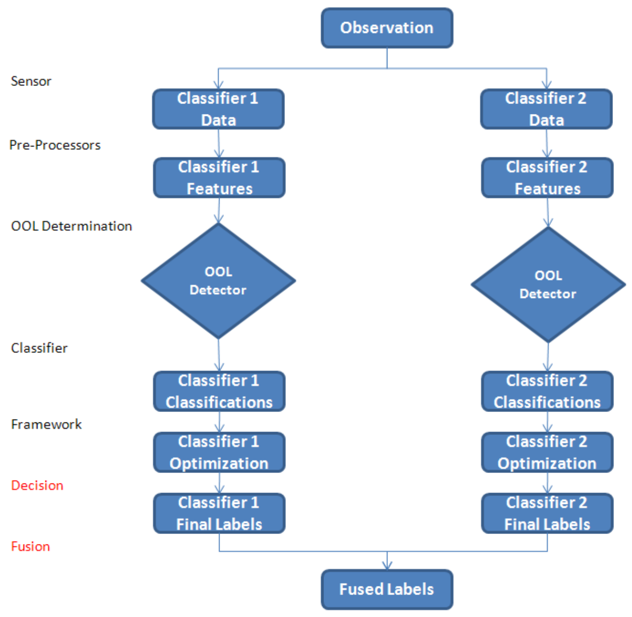

Figure 5.9 Classification System within mathematical optimization framework. In this process two independent classifiers operate on sensor data and produce their individual optimal outputs. These outputs are then fused to produce the final system output.

Under the local optimization model, the Hybrid Classifier operating on the HH polarized data, optimal parameter settings are chosen within the framework so as to optimize a certain performance, such as $CA_{avg}$. This is then repeated for the

VV polarized data. Each of the four fusion methods are then used to combine the optimal results from each sensor to produce a *combined* optimal result.

Model 2 is the global optimization model, which is shown in Figure 5.10. Under the global optimization model, we first fuse the HH and VV outputs for the various parameter settings, then determine the optimal parameter settings.

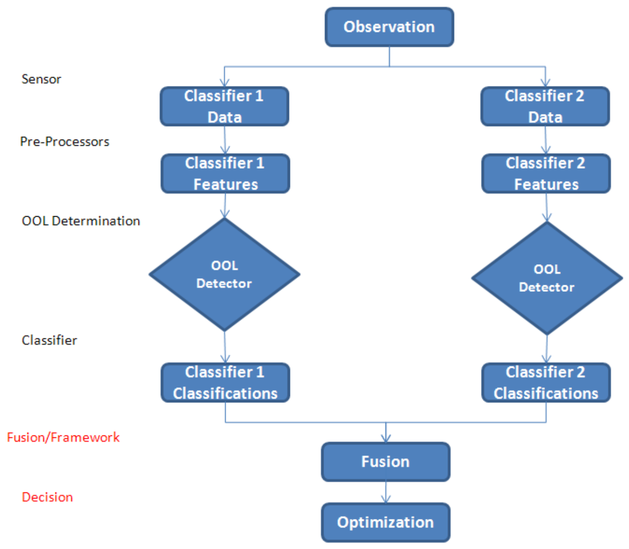

Figure 5.10 Classification System within mathematical optimization framework. In this process two independent classifiers are fused within the framework, thus producing the best overall parameter settings for the fused classification system.

We use the framework to select the thresholds that maximize each of our choices for parameter settings. The formulation becomes:

**Objective Function**

$$\max_{x \in X} CA_{avg}(x)$$

**Subject to:**

**Parameter Constraints**

$$\begin{aligned}
\sum_{i=1}^{q} Q_i &= 1 && \text{use one of the } q \text{ quantile methods} \\
\sum_{i=1}^{w} W_i &= 1 && \text{use one of the } w \text{ wedge sizes} \\
\sum_{i=1}^{n} \theta_{1i} &= 1 && \text{use one of the } n \text{ noise thresholds} \\
\sum_{i=1}^{s} M_i &= 1 && \text{use one of the } s \text{ similarity metrics} \\
\sum_{i=1}^{f} F_i &= 1 && \text{use one of the } f \text{ fusion methods} \\
\sum_{i=1}^{o} O_i &= 1 && \text{use one of the } o \text{ optimization methods}
\end{aligned}$$

### *5.3.2 Forced Decision Experimental Results*

For our experiments, we list the optimal parameter settings and the associated MOEs under the mathematical framework for both the local optimization model and the global optimization model in Table 5.3. Under each model scenario, we optimize $CA_{avg}$ as the objective function and boldface the optimal result under each model in Table 5.3. For each optimum we include the results for the other MOEs and the two respective error rates. We also include the individual optimal results from the individual sub-optimization problems of Model 1 in the results listed in Table 5.3. In model 1, the PNN fusion method achieves the highest result for $CA_{avg}$, while in Model 2 the BEM produces the highest $CA_{total}$. Aside from the fusion method, the other parameter settings that change when optimizing in Model 2 versus Model 1 are the similarity metric and number of quantiles; while in both cases the noise threshold and wedge size stayed the same.

Table 5.3 Hybrid Classifier Results: Forced Decision.

| **Local Optimization Model** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Parameters** | | | | | **MOEs** | | | | | | | |
| $Q$ | $\theta_1$ | $W$ | $M$ | $F$ | $CA_{avg}$ | $LA_{avg}$ | $CA_H$ | $LA_H$ | $CA_{FN}$ | $LA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ |
| 5 | 0.5 | 7 | Euclidean | HH Data | 0.679 | 0.685 | 0.874 | 0.855 | 0.861 | 0.862 | 0.137 | 0.183 |
| 5 | 0.5 | 7 | Euclidean | VV Data | 0.684 | 0.690 | 0.867 | 0.873 | 0.862 | 0.860 | 0.130 | 0.187 |
| Fused | 0.5 | 7 | Euclidean | BEM | 0.698 | 0.717 | 0.924 | 0.860 | 0.849 | 0.848 | 0.113 | 0.189 |
| Fused | 0.5 | 7 | Euclidean | Borda | 0.687 | 0.710 | 0.936 | 0.833 | 0.812 | 0.812 | 0.126 | 0.187 |
| Fused | 0.5 | 7 | Euclidean | PNN | **0.733** | 0.737 | 0.918 | 0.899 | 0.897 | 0.897 | 0.092 | 0.175 |
| Fused | 0.5 | 7 | Euclidean | Bayes | 0.722 | 0.724 | 0.891 | 0.887 | 0.887 | 0.887 | 0.111 | 0.167 |

| **Global Optimization Model** | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Parameters** | | | | | **MOEs** | | | | | | | |
| $Q$ | $\theta_1$ | $W$ | $M$ | $F$ | $CA_{avg}$ | $LA_{avg}$ | $CA_H$ | $LA_H$ | $CA_{FN}$ | $LA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ |
| 12 | 0.5 | 7 | Manhattan | BEM | **0.768** | 0.777 | 0.922 | 0.892 | 0.888 | 0.919 | 0.123 | 0.123 |
| 11 | 0.5 | 7 | Manhattan | Borda | 0.713 | 0.732 | 0.948 | 0.834 | 0.811 | 0.811 | 0.120 | 0.167 |
| 8 | 0.5 | 7 | Manhattan | PNN | 0.745 | 0.762 | 0.917 | 0.888 | 0.884 | 0.884 | 0.099 | 0.156 |
| 11 | 0.5 | 7 | Manhattan | Bayes | 0.736 | 0.738 | 0.890 | 0.882 | 0.881 | .0889 | 0.114 | 0.149 |

To further display the results reported in Table 5.3, we also present the associated confusion matrices for the optimal result from each model. The confusion matrices for the the two models are presented in Tables 5.4 and 5.5. The entries in each cell of the confusion matrices are given in terms of number of occurrences for each target/label combination.

Table 5.4 Hybrid Classifier Results: Forced Decision, Model 1, PNN.

| **C ↓ L →** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **285** | 26 | 1 | 4 | 6 | 17 | 2 | 4 | 10 | 5 |
| 2 | 12 | **323** | 6 | 5 | 8 | 3 | 2 | 1 | 0 | 0 |
| 3 | 0 | 14 | **240** | 15 | 69 | 7 | 0 | 14 | 1 | 0 |
| 4 | 0 | 13 | 37 | **285** | 31 | 6 | 21 | 21 | 2 | 0 |
| 5 | 0 | 11 | 75 | 14 | **229** | 5 | 3 | 22 | 1 | 0 |
| 6 | 23 | 13 | 6 | 7 | 5 | **257** | 1 | 3 | 7 | 38 |
| 7 | 6 | 1 | 2 | 2 | 3 | 0 | **278** | 45 | 22 | 1 |
| 8 | 0 | 7 | 7 | 18 | 20 | 0 | 69 | **231** | 8 | 0 |
| 9 | 14 | 1 | 8 | 3 | 13 | 3 | 26 | 18 | **266** | 8 |
| 10 | 20 | 1 | 3 | 0 | 2 | 24 | 3 | 1 | 5 | **301** |

Table 5.5 Hybrid Classifier Results: Forced Decision, Model 2, BEM.

| **C ↓ L →** | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **267** | 23 | 6 | 4 | 10 | 25 | 1 | 4 | 13 | 7 |
| 2 | 10 | **306** | 11 | 17 | 9 | 4 | 3 | 0 | 0 | 0 |
| 3 | 0 | 4 | **264** | 9 | 62 | 3 | 1 | 15 | 2 | 0 |
| 4 | 0 | 12 | 30 | **248** | 33 | 0 | 10 | 26 | 1 | 0 |
| 5 | 0 | 7 | 62 | 20 | **249** | 0 | 1 | 15 | 6 | 0 |
| 6 | 9 | 16 | 11 | 10 | 9 | **274** | 1 | 5 | 7 | 18 |
| 7 | 4 | 2 | 5 | 3 | 3 | 2 | **295** | 37 | 8 | 1 |
| 8 | 0 | 6 | 13 | 8 | 17 | 0 | 27 | **284** | 5 | 0 |
| 9 | 12 | 2 | 6 | 1 | 17 | 3 | 16 | 15 | **286** | 2 |
| 10 | 13 | 7 | 5 | 2 | 13 | 19 | 2 | 0 | 13 | **286** |

Our results indicate increased performance across all fusion methods. In every case, $CA_{total}$ is larger when optimizing post fusion, rather than fusing the two sub-optimization results. To test the validity of these results, we present a test for statistical significance.

### *5.3.3 Test for Statistical Significance: Forced Decision Experiments*

We first introduce some theory behind our testing [70]. We conduct experiments from two different models, the Local Optimization and the Global Optimization. The structural difference between the two is the order in which fusion and optimization occur. For each model, we have several MOEs. Here, we choose to use $CA_{total}$. Welch [70] points out that testing $H_0 : \theta_A = \theta_B$ is equivalent to forming the confidence interval

$$\widehat{\theta_A} - \widehat{\theta_B} \pm SE(\widehat{\theta_A} - \widehat{\theta_B})C(\alpha)$$

and seeing if the confidence interval contains 0, where $\theta_A$ is the output from model A and $\theta_B$ is the output from model B. For our tests, both models use the same data, thus, we have non-independent replications. We form a paired T-Test by

$$V_n = \theta_{A_n} - \theta_{B_n}$$

where $n = 1, 2, ..., N$ and $N$ is the number of replications. Then we can use the $t$ statistic

$$t = \frac{\overline{V}}{\sqrt{S_V^2/N}} \sim t_{N-1,\alpha}$$

where $\overline{V}$ is the average difference between the outputs of the two models over all replications and $S_V^2$ is the sample variance of $V$ over all replications.

For our Forced Decsion Experiments, we use the initial results to guage our parameter settings. The parameter settings for both the Local Model and the Global Model are given in Table 5.6. Both models are then replicated 30 times, where for each replication we randomize the data, so that each replication will have different template, test and validation sets.

Table 5.6 Hybrid Classifier: Forced Decision Parameter Settings.

| Model | $Q$ | $\theta_1$ | $W$ | $M$ | $F$ |
|---|---|---|---|---|---|
| Local | 5 | 0.5 | 7 | Euclidean | PNN |
| Global | 12 | 0.5 | 7 | Manhattan | BEM |

For the 30 replications of the Forced Decision optimization, we get $\overline{V} = 0.094$ and $S_V^2 = 0.001$. Using $\alpha = 0.05$, we have

$$t = \frac{0.094}{\sqrt{0.001/30}} = 14.843 > 1.699 = t_{29,0.05},$$

thus, since $t > t_{29,0.05}$, we can conclude that there is a significant difference between the Global and Local models under a Forced Decison.

## *5.4 Implementing the Hybrid Classifier with a NDEC Option*

As stated in [12], in all classification problems, decisions must be made which effect the overall quality of the classification system. Decision makers may impose constraints on a classifier due to their own willingness to risk critical and/or non-critical errors. This leads to the possibility of non-declarations. For a given exemplar, the choice to make a classification decision is usually based on thresholding the measure the classifier employs to make decisions. Such measures include, but are not limited to the Minkowski metric used by the hybrid classifier and the squared Mahalanobis distance used by the feature vector classifier. These thresholds create a rejection region which encompasses a certain interval of measures for which a classification decision is not made. The rejection region thus allows for a classification label when a classifier output falls outside the rejection region and disallows classification labels when a classifier output falls within the region. Previous non-declaration methods have included Chow [8] who stated that classification accuracy can be improved by withholding label assignments for exemplars which are difficult to classify. Chow's work used an optimal rule for rejection based on a single threshold for the posterior probability of a given class. For $N$ classes, do not make a classification decision for exemplar $x$ if the (winning) posterior probability for class $i$ given $x$, $P(\omega_i|x)$ is less than the threshold $T$. We can express this optimal rejection rule as: given $T \in (0, 1]$,

$$\max_{k \in \{1,2,\ldots,N\}} P(\omega_k|x) = P(\omega_i|x) < T. \tag{5.1}$$

Fumuera et al. [15] showed that Chow's work could be improved by allowing for by-class thresholds, rather than a single threshold for all classes. They based this improvement when they noted that if assumption of perfect knowledge of posterior probabilities were violated, no single threshold value could be used to find an optimal decision threshold. Using their scheme, a classification decision for exemplar $x$ is not made if

$$\max_{k=1,2,\ldots,N} \widehat{P}(\omega_k|x) = \widehat{P}(\omega_i|x) < \theta_i \tag{5.2}$$

where $\widehat{P}(\omega_i|x)$ is the estimated posterior probability for class $i$ given $x$.

Laine [34] designed an optimization framework that maximized the probability of a true positive declaration while meeting warfighter constraints that were concerned with setting upper bounds on errors while setting lower bounds on declaration rate. Albrecht [3] extending this framework to include constraints on out-of-library performance. Friend [12] extended both of these previous works by improving the methodologies for non-declarations. His research employed entropy and Kullback-Liebler distance as methods for identifying exemplars for which insufficient evidence for classification decisions exist.

The feature vector method of Friend [12] processes data in the following manner. For the feature vector method, a given HRR profile exemplar's range is first cropped to only include the middle portions. This was introduced by Mitchell [46]. We will use Friend's user-defined range length of 120, thus we have a bin window width of 12 for each of the 10 range bin windows. The feature vector for a given exemplar will then be the maximum amplitudes within each of the 10 range bins. Similar to the hybrid classifier, the feature vector classifier forms templates for each class from each of the 360 exemplars in the training set. Classification decisions are made by computing the Mahalanobis distance from the exemplar to each of the 10 classes as follows. Let $x_\alpha$ be a feature vector for a given exemplar at a particular aspect angle $\alpha$. The Mahalanobis distance, $M_{\alpha,i}$, for that given exemplar at aspect angle $\alpha$ from class $i$ is given by

$$M_{\alpha,i} = (x_\alpha - \widehat{\mu}_{\alpha,i})' \widehat{\Sigma}_{\alpha,i}^{-1} (x_\alpha - \widehat{\mu}_{\alpha,i}). \tag{5.3}$$

We extend the previous works by improving upon these methods. The idea of a single non-declaration threshold per class introduced in Fumera et al. [15] and implemented most recently by Friend [12] is extended by determining non-declaration status within the current exemplar itself. For this current non-declaration scheme, a classification decision for exemplar $x$ is not made if

$$\max_{k \in \{1,2,\dots,N\}} \widehat{S}(\omega_k|x) = \widehat{S}(\omega_i|x) < \theta_{i,\alpha} \tag{5.4}$$

where $\widehat{S}(\omega_i|x)$ is the estimated similarity measure for class $i$ given $x$ at aspect angle $\alpha$.

Further, we note that difficulty in declarations is not simply due to the value of the winning score, but is largely due to the difficulty in distinguishing between at least two different classes. Thus, rather than simply thresholding on the single winning score, we will threshold on the difference between the class with the winning score and the class with the next closest score. Thus we have

$$\max_{k\in\{1,2,\ldots,N\}} \widehat{S}(\omega_k|x) - \max_{i\neq k\in\{1,2,\ldots,N\}} \widehat{S}(\omega_k|x) < \theta_{i\alpha} \tag{5.5}$$

where $\theta_{i\alpha}$ is some percentage of the overall range of scores for that exemplar.

For our initial non-declaration experiments, we will use the same 10 in-library targets as in the forced decision experiments. Now, we will allow for non-declarations using the following method. For a given exemplar, a non-declaration is made if the distance between the winning class' score and the next closest class' score is less than some percentage of the overall range of scores.

### *5.4.1 NDEC Experimental Results*

With a non-declaration option in place, the hybrid classifier operates on one of the two sensor data sets, the HH or VV polarized data using the full HRR profile for each exemplar. Within the framework, each individual classifier/sensor combination is optimized. Under the Local Model, the Hybrid Classifier operating on the HH polarized data, optimal parameter settings are chosen within the framework so as to optimize a certain measure of performance, such as $CA_{total}$. This is repeated for the VV polarized data. Each of the four fusion methods are used to combine the optimal results from each sensor to produce a combined optimal results. Under the Global

Model, we first fuse the HH and VV outputs for the various parameter settings, then determine the optimal parameter settings. We use the framework to select the thresholds that maximize each model across our choices for parameter settings. We now allow the non-declaration option, where the non-declaration threshold is chosen from the set $\{0.00, 0.01, 0.02, 0.03\}$. The mathematical framework formulation becomes:

**Objective Function**

$$\max_{x \in X} CA_{total}(x)$$

**Subject to:**

**Parameter Constraints**

$$\begin{aligned}
\sum_{i=1}^{q} Q_i &= 1 && \text{use one of the } q \text{ quantile methods} \\
\sum_{i=1}^{w} W_i &= 1 && \text{use one of the } w \text{ wedge sizes} \\
\sum_{i=1}^{n} \theta_{1i} &= 1 && \text{use one of the } n \text{ noise thresholds} \\
\sum_{i=1}^{s} M_i &= 1 && \text{use one of the } s \text{ similarity metrics} \\
\sum_{i=1}^{f} F_i &= 1 && \text{use one of the } f \text{ fusion methods} \\
\sum_{i=1}^{o} O_i &= 1 && \text{use one of the } o \text{ optimization models}
\end{aligned}$$

**Non-Declaration Constraint**

$$\sum_{i=1}^{d} D_i \quad = \quad 1 \qquad \text{use one of the } d \text{ non-declaration thresholds}$$

Table 5.7 shows the optimized results for each model. Under each model scenario, we optimize $CA_{total}$ as the objective function. For each optimum we include the results for the other MOEs and the two respective error rates. We also include the declaration rate, DecRate, which reports the percentage of exemplars for which the classifier makes a class labeling decision. We also include the individual optimal results from the individual sub-optimization problems of model one in the results listed in Table 5.7.

As in the forced decision experimental results, we present the associated confusion matrices for the optimal result from each model. The confusion matrices for the the two models are presented in Tables 5.8 and 5.9. The entries in each cell of the confusion matrices are given in terms of number of occurrences for each target/label combination.

### *5.4.2 Test for Statistical Significance: NDEC Option*

For our NDEC Experiments, we again use the initial results to guage our parameter settings for 30 replications. The parameter settings are given in Table

5.10. For our replications, we randomize the data each time, so that each replication will have different template, test and validation sets.

For the 30 replications of the NDEC optimization, we get $\overline{V} = 0.149 \& S_V^2 = 0.001.$. Using $\alpha = 0.05$, we have

$$t = \frac{0.149}{\sqrt{0.001/30}} = 23.092 > 1.699 = t_{29,0.05},$$

thus, sicne $t > t - 29.0.05$ we can conclude that there is a significant difference between the Global and Local models with the NDEC option.

## *5.5 Implementing the Hyrid Classifier with OOL Targets*

Thus far, we have demonstrated the ability of our classifiers to correctly classify objects for which they have been trained by presenting the classifiers with known representations of those objects. In many cases, a complete set of possible objects is not known a priori. One method for overcoming this obstacle is to use the NDEC labeling for objects where insufficient information exists to make a classification decision. Another growing trend in ATR classification is to provide a label for objects for which the classifier is not trained to recognize [3, 12]. Such objects will be labeled out-of-library (OOL).

### *5.5.1 OOL Background*

Friend [12] uses the following out-of-library procedure for the feature vector classifier. Let the class label for exemplar $x$ be denoted $L_x$ where $L_x \in \{1, ..., 10\}$. For exemplar $x$, the class label, $L_x$, is determined by finding the class corresponding to the minimum average Mahalanobis distance. $L_x = \min dM_{x,d}$ and $M_x = \min_d M_{x,d}$. By looking across all exemplars in a training set by class for only those exemplars that were correctly classified, we can determine the maximum correct average Mahalanobis distance. Friend [12] actually quantizes the correct average Mahalanobis distances, but for simplification, we will only consider the maximum. This maximum value is used as a class specific threshold, $T_i = \max_d M_x$. Then, an exemplar is given out-of-library status if the minimum Mahalanobis distance corresponds to class $i$ and $M_x > T_i$. Otherwise, it is considered an in-library exemplar and classified as one of the in-library classes or given non-declaration status.

Table 5.11 lists the five OOL targets used in Friend's research. According to the Friend methodology, a specific exemplar is deemed to be in-library if the corresponding average Mahalanobis distance is less than the class specific threshold, and one has a complete absence of confidence that an exemplar is in the library otherwise. This is nothing more than a confidence step function. Thus, in this method, an exemplar is only observed by the IL classifier if the out-of-library system has complete confidence that the exemplar is in-library. The out-of-library confidence values as implemented in this section are binary values in the set $\{0, 1\}$. This methodology has been previously replicated and reported in Leap [36]. These results are in Table 5.12.

Table 5.12 reports results for three different measures of performance. The first is the True Positive Rate (TPR) for hostile targets. That is the probability that the classifier assigns a hostile label to any hostile exemplar. So, for our data set, if an exemplar comes from any of classes 1,2,...,5, and if the label assigned to the exemplar is one of labels 1,2,...,5, this exemplar is a true positive. Friend [12] details the formal computation of the TPR. The next performance quantifier is the average classification average for all in-library classes (IL CA). For this computation, the classification accuracy for each of the 10 in-library targets is computed. In the case of the in-library targets this computation is done for a specific class, $i$ by dividing the total number of exemplars from that class that are correctly classified, $TC_i$ by the total number of exemplars of that class, $N_i$. However, as outlined by Friend [12], the total number of exemplars $N_i$ is reduced when employing NDEC and OOL methods. The OOL Detector must first determine the number of exemplars that should be presented to the classifier. Thus, the total number of exemplars of the given class type is reduced to $N_i - OOL_i$. Next, the classifier will make classification determinations for those exemplars that pass the previously described NDEC criteria. For the case where some exemplars do not, the total number of exemplars of the given class type is reduced again, leaving a total of $N_i - OOL_i - NDEC_i$ exemplars to be classified. Thus, the average classification accuracy for a given class is $\frac{TC_i}{N_i - OOL_i - NDEC_i}$. The out-of-library classification accuracy (OOL CA) is the number of out-of-library exemplars that are correctly identified as out-of-library by the OOL Detector $C_{OOL}$, divided by the total number of out-of-library exemplars, $N_{OOL}$, which is $\frac{C_{OOL}}{N_{OOL}}$.

The OOL Quantile Method of Friend [12] is as follows. Using the training data set, the squared Mahalanobis distance scores for all correctly identified exemplars of each class type are identified. The maximum squared Mahalanobis distance corresponding to a correctly identified exemplar is identified as the OOL threshold for the class. Friend allows for any quantile to be used; however, he reports results for the maximum score, which we implement.

### *5.5.2 OOL Methodology Improvements*

Friend [12] and Leap [36] both note common trends in the feature vector template method. Friend [12] noted that classification accuracy over all targets decreased dramatically over certain aspect angles ($\sim 90°$ and $\sim 270°$). Leap [36] noted a marked decrease in OOL classification accuracy as the number of looks at a given target increased. Table 5.12 shows that the Friend OOL detector has a significant decrease in performance when the number of looks at a target increases from five to ten. Leap postulated that this decrease in OOL classification accuracy was due to the heterogeneous nature of the HRR profiles. The OOL method of Friend [12] determines the winning template for each look by choosing the template which corresponds to the minimum squared Mahalanobis distance from the test exemplar. When more than one look was used, the average Mahalanobis distance over all looks is computed and the winning template is the template with the smallest average Mahalanobis distance across all looks. The heterogeneous nature of the data allows the true class' Mahalanobis distance to become inflated, while allowing incorrect class' Mahalanobis distances to fall within the OOL threshold learned in the training phase.

We improve the OOL methodology by first breaking up each class into equally sized windows across all aspect angles. For example, in the results which follow we use 24 aspect angle windows, each being 15 degrees in width. We then employ two separate independent OOL detectors. The first OOL detector is a self-organized Kohonen map. The second is a Probabilistic Neural Network (PNN). Previous methods [12,36] have attempted to use neural networks, such as the PNN as an independent OOL detector. In these cases, the main thrust was to train the OOL detector in a 2-class problem, where the two classes consisted of either in-library or out-of-library targets. Leap's effort attempted to exploit the entire feature space, noting that certain regions of the feature space exist where there are no in-library targets. This observation led to the attempt to use these regions as the features for the out-of-library class when training the PNN. Leap's research went on to bound and discretize the feature space, next determining the Mahalanobis distance from each discrete point to the in-library class. If this distance from the discrete point to the in-library class exceeds a user defined threshold, then the discrete point is deemed to be a member of the out-of-library class. Otherwise, it is deemed an in-library point. This is repeated for each discrete point in the feature space. Once the entire discretized feature space has been divided into in-library and out-of-library regions, these points can be used to train a generalized regression neural network (GRNN). While this methodology works well on a toy problem of multivariate normal data, this new OOL methodology performs poorly on the DCS data set. The main reason for this is simple. The OOL targets occupy the same regions of the feature space as do the IL targets. Thus, an OOL target exemplar is very likely to be classified as belonging to one of the in-library classes.

### *5.5.3 Artificial Neural Networks as OOL Detectors*

The main idea of the Leap OOL Detector, that of finding a reasonably low dimension feature space which could be discretized, inspires the development of our OOL Detectors. To overcome the feature space dilemma, we propose the following. Rather than grouping all in-library data as belonging to a single class, we will treat each in-library target separate from the rest of the in-library data. Thus, when we train the Kohonen map or the PNN, we will actually develop an OOL detector for each IL target. To do so, we present the training data as follows. Each IL target is divided into its own class for training. The remaining nine in-library classes are grouped together and used as the OOL class for training. We then train the OOL detector specific to each target to only recognize the difference between that target class and anything else. If a test exemplar is close enough to the target class, the OOL detector deems it in-library. Otherwise, it is labeled OOL. To successfully utilize the new OOL detectors, we will make one assumption. That is, we assume that the classification system has already pre-screeened a test exemplar and made a determination that the exemplar belongs to a certain target class. This is done using the forced decision methodology presented earlier. Thus, the OOL detector will assume that the classifier is 100 percent accurate in identifying in-library targets which it has been trained to recognize. So, we have narrowed down the work for each target OOL detector by only asking it to recognize a single in-library target, while rejecting as OOL any exemplar not belonging to its own class.

### *5.5.4 Self Organizing Maps*

A self-organizing map (SOM) is a type of artificial neural network that is trained using unsupervised learning to produce a low-dimensional (typically two dimensional), discretized representation of the input space of the training samples, called a map. The map seeks to preserve the topological properties of the input space. This makes SOM a useful tool for creating low-dimensional views of high-dimensional data. The model was first described as an artificial neural network by the Finnish professor Teuvo Kohonen [31], and is often referred to as a Kohonen map. Figure 5.11 shows the architecture for a Kohonen map.

A self-organizing map consists of components called nodes or neurons. Associated with each node is a weight vector of the same dimension as the input data vectors and a position in the map space. The usual arrangement of nodes is a regular spacing in a hexagonal or rectangular grid. The self-organizing map describes a mapping from a higher dimensional input space to a lower dimensional map space. The procedure for placing a vector from the data space onto the map is to find the node with the closest weight vector to the vector taken from data space and to assign the map coordinates of this node to our vector. While it is typical to consider this type of network structure as related to feed forward networks where the nodes are visualized as being attached, this type of architecture is fundamentally different in arrangement and motivation.

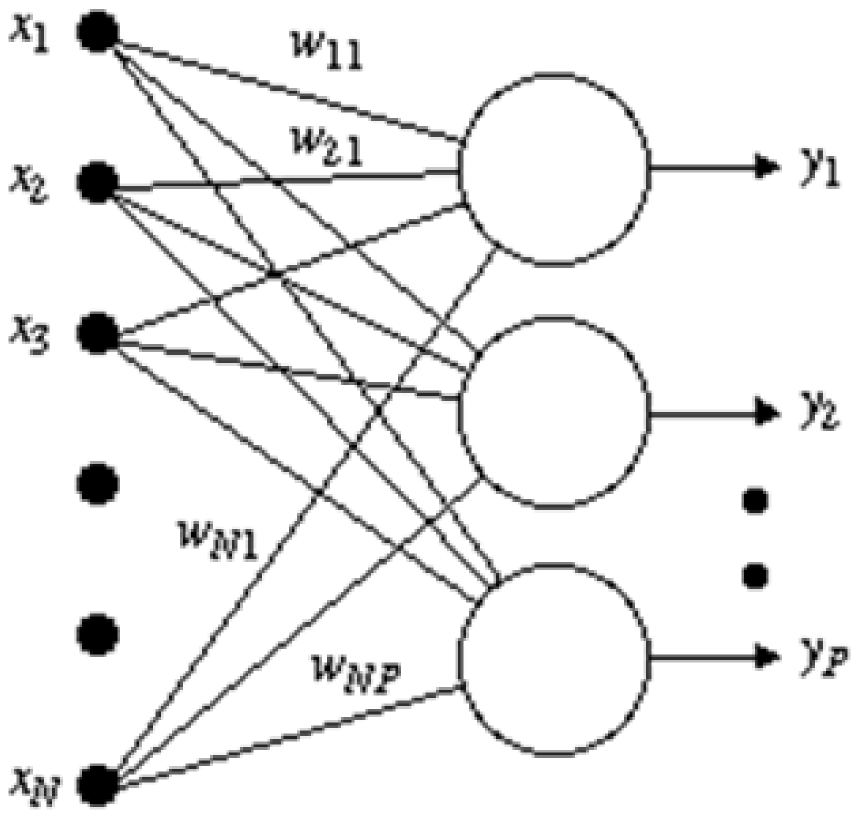

Figure 5.11 SOM Architecture [41].

The goal of learning in the self-organizing map is to cause different parts of the network to respond similarly to certain input patterns. The weights of the neurons are initialized either to small random values or sampled evenly from the subspace spanned by the two largest principal component eigenvectors. The network must be fed a large number of example vectors that represent, as close as possible, the kinds of vectors expected during mapping. The examples are usually administered several times. The training utilizes competitive learning. When a training example is fed to the network, its Euclidean distance to all weight vectors is computed. The neuron with weight vector most similar to the input is deemed to be the winning neuron. The weights of the winning neuron and neurons close to it in the SOM lattice are adjusted towards the input vector. The magnitude of the change decreases with time and with distance from the winning neuron. The update formula for a neuron with weight vector, $W(t)$ is:

$$W(t+1) = W(t) + \Theta(v,t)\alpha(t)\left(D(t) - W(t)\right)$$

where,

| | | |
|---|---|---|
| $W(t)$ | $=$ | weight vector at time $t$ |
| $\Theta(v,t)$ | $=$ | the neighborhood function for the given SOM lattice |
| $\alpha(t)$ | $=$ | a monotonically decreasing adaptation parameter |

$D(t) \quad = \quad$ the input distance vector.

The Kohonen map reflects the inner structure of the training data. However, one cannot say which neurons are activated by which input vectors. In addition, the neurons corresponding to some input vectors after a particular training, will correspond to another set of vectors after another training run. So the SOM has to be calibrated. This can be achieved by presenting well known examples to the net and by recording which neuron is activated with a given example vector. As Kohonen maps tend to form some kind of elastic surface on the range of input vectors of the training data, neurons which are not activated in the calibration process may be interpreted by interpolation.

In developing the SOM into an OOL Detector, we follow the following procedure. For each in-library target class, a training data set is broken up into two classes. The first class is the target of interest, which we will call the TOD. The remaining in-library classes are used as the second class, which we use as the OOL class for the purpose of training the SOM. Using a 4 x 4 rectangular grid, the training data is then divided into 24 wedges, according to aspect angle, each ranging $15°$ in width. After training is completed, the SOM OOL detector then has a map according to the following scheme. Each in-library target contains 24 maps, where each map corresponds to a $15°$ width of aspect angle. To test the SOM, recall that in the composite classification system, for a given test exemplar $x_i$, the classifier, $C^1_{d,\theta_1}$ yields an output, $L_i$ which is a labeling assignment to one of the template classes, all of which are in-library. This labeling output is then used as an input to the OOL Detector of the corresponding in-library class, where the assumed aspect of the exemplar $x_i$ is used to pick the correct aspect angle wedge of that class' OOL Detector. Clearly, if the classifier makes an incorrect labeling assignment, the choice of OOL Detector will be incorrect. Thus, the output from the OOL Detector will likely be incorrect as well. However, to test the utility of the SOM as an OOL Detector, we conduct an independent test. For this experiment, we train the SOM OOL Detector as we have described using a training set consisting of one TOD class and the remaining IL targets as the second class. The test set presented to the trained SOM is then made up of a test set of TOD data, not previously used during training and a second set of test data, which consists of the five OOL classes presented in Table 5.11. In order to test the SOM OOL Detector independent of the Hybrid Classifier, we assume that all labeling assignments presented to the SOM OOL Detector are correct. The results of the SOM OOL Detector experiment are shown in Table 5.13.

For this experiment, the SOM OOL has a $CA_{total} = 0.7049$. In subsequent experiments, we will combine the SOM OOL Detector with the Hybrid Classifier, which we describe in more detail later. We note similar problems as previous classifiers according to the aspect angle of the exemplar to be classified. Figure 5.12 shows the SOM OOL Detector performance by aspect angle.

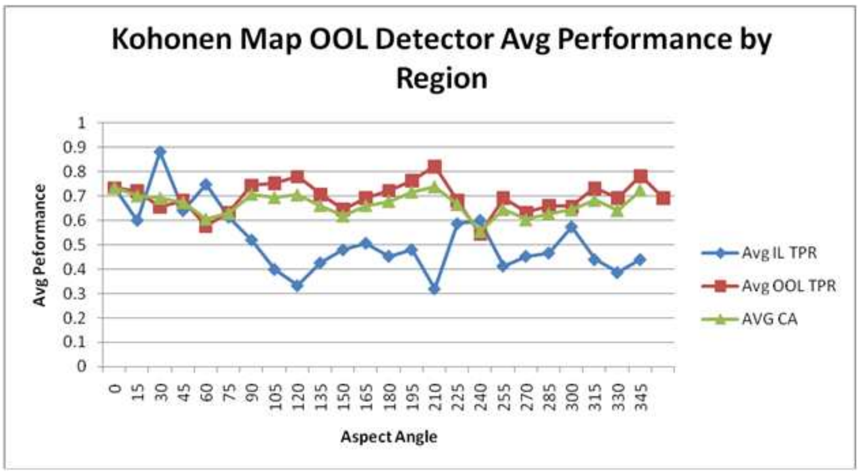

Figure 5.12 SOM OOL Detector Performance by Region.

### *5.5.5 Probabilistic Neural Networks*

Wasserman [68] describes how the PNN operates as a classifier as follows. For each class, a Gaussian probability density function (PDF) is placed around each data point in the training set of that class. All PDFs are then added and then normalized. The normalized input vector $X = (X_1, X_2, ..., X_n)$ is applied to the distribution layer neurons.

This layer does not perform any computations, but merely serves as a connection point. Each training vector is used to calculate a set of weights, where each weight has the value of a component of that vector. Pattern layer neurons are grouped by the known classification of its associated training vector. Each pattern layer neuron sums the weighted inputs from every distribution layer neuron [68]. This is equivalent to taking the sum of squares of the training set and the test set, $(X - X_{R,i})^T(X - X_{R,i})$, where $X_{R,i}$ is the $i$th exemplar in the $R$th class from the

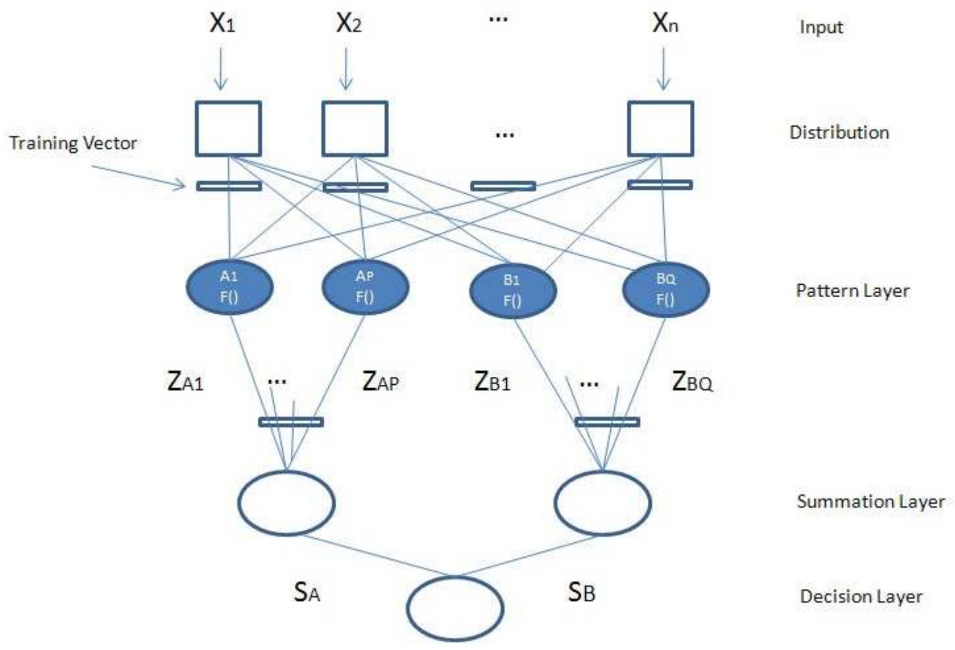

Figure 5.13 Probabilistic Neural Network [68].

training set. From normalization, this reduces to $(X_{R,i}^T X_i - 1)$. The pattern layer neurons then apply a non-linear function to the corresponding sum producing an output $Z_{c,i}$, where $c$ indicates the true class of the training vector and $i$ indicates the pattern layer neuron. The non-linear function for $Z_{c,i}$ is

$$Z_{c,i} = \exp\left(\frac{(X_{R,i}^T X_i - 1)}{\sigma^2}\right). \tag{5.6}$$

In this equation, $X$ is defined above and the set of weights corresponding to a pattern neuron represent a training vector $X_{R,i} = (X_{R,1}, X_{R,2}, \ldots, X_{R,n})$. The summation layer simply sums the $Z_{c,i}$ for each class [68]. Thus, the output of the summation layer for a specific class, $S_c$ is

$$S_c = \sum_{i=1}^{n} \exp\left(\frac{(X_{R,i}^T X_i - 1)}{\sigma^2}\right). \tag{5.7}$$

The decision layer compares $S_c$ for all classes and assigns the input vector to the class with the largest corresponding $S_c$.

In developing the PNN into an OOL Detector, we follow the same procedure as with the SOM OOL Detector. For each in-library target class, a training data set is broken up into two classes. The first class is the target of interest, which we will call the TOD. The remaining in-library classes are used as the second class, which we use as the OOL class for the purpose of training the SOM. The training data is then divided into 24 wedges, according to aspect angle, each ranging 15° in width. After training is completed, the PNN OOL Detector has then has been trained according to the following scheme. Each in-library target contains 24 maps, where each map corresponds to a 15° width of aspect angle. We conduct the same independent test for the PNN OOL Detector as for the SOM OOL Detector. For this experiment, we train the PNN OOL Detector as we have described using a training set consisting of one TOD class and the remaining IL targets as the second class. The test set presented to the trained SOM is then made up of a test set of TOD data, not previously used during training and a second set of test data, which consists of the five OOL classes presented in Table 5.11. In order to test the SOM OOL Detector independent of the Hybrid Classifier, we assume that all labeling assignments presented to the PNN OOL Detector are correct. The results of the PNN OOL Detector experiment are shown in Table 5.14.

For this experiment, the PNN OOL has a $CA_{total} = 0.8063$. In subsequent experiments, we will combine the PNN OOL Detector with the Hybrid Classifier, which we describe in more detail later. We note similar problems as previous classifiers according to the aspect angle of the exemplar to be classified. Figure 5.14 shows the PNN OOL Detector performance by aspect angle.

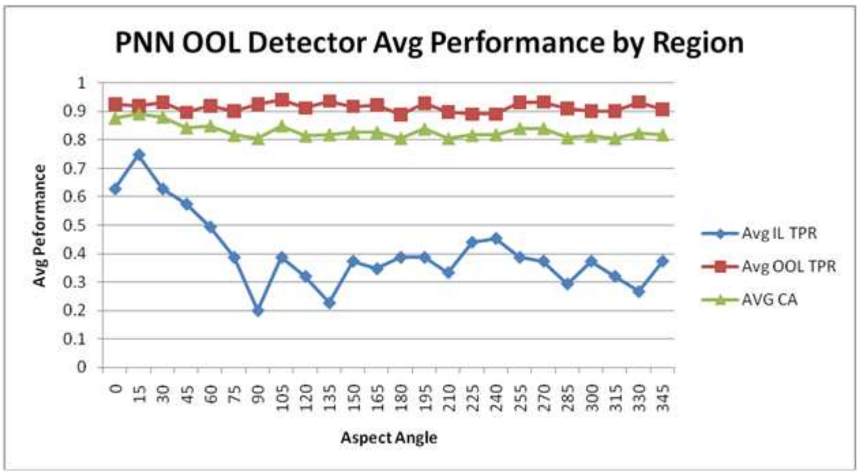

Figure 5.14 PNN OOL Detector Performance by Region.

### 5.5.6 OOL Detector Summary

Two important notes we should empahsize concerning the SOM and PNN OOL Detectors are the following. First, both OOL Detectors can operate on any features taken from the original data. We can use as many or as few as we choose without having to make dramatic changes to the overall algorithmic structure of either OOL Detector. The test results we reported here were actually confirmed by changing the number of features several times, with little change in the performance of the OOL detectors. Second, and a key point, is the fact that both OOL Detectors fall under the definition of blind methods. Recall the definition of a blind method is the following.

A *Blind* Method is based solely on in-library training data without any knowledge of test data class membership. OOL criteria is based entirely on in-library training data and the characteristics of a test exemplar.

We have thus developed two independent OOL Detectors that operate independently of the classifier from which they receive inputs, meaning they are not a heuristic search technique embedded within a classifier. Moreover, they do not create their decision thresholds from any information other than training data. Thus, our OOL Detectors operate under the most difficult of conditions, while producing very favorable results.

We now demonstrate the utility of the two independent OOL detectors by using them in conjunction with the already tested Hybrid Classifier. By allowing the OOL Detectors to operate independent of the classifier, a more true OOL decision can be expected, where classifier performance does not influence OOL Detection rates, as observed in previous research [12, 36].

### *5.5.7 OOL NDEC Experimental Results*

With a non-declaration option in place, the hybrid classifier operates on one of the two sensor data sets, the HH or VV polarized data using the full HRR profile for each exemplar. We now present each classifier with the same ten IL targets as in previous experiments, as well as the five OOL targets discussed earlier. Within the framework, each individual classifier/sensor combination is optimized. Under the Local Model, the Hybrid Classifier operating on the HH polarized data, optimal parameter settings are chosen within the framework so as to optimize a certain measure of performance, such as $CA_{total}$. This is then repeated for the VV polarized data. Each of the four fusion methods are then used to combine the optimal results from each sensor to produce a combined optimal results. Under the Global Model, we first fuse the HH and VV outputs for the various parameter settings, then determine the optimal parameter settings. We use the framework to select the thresholds that maximize each of across our choices for parameter settings. We now allow the non-declaration option, where the non-declaration threshold is chosen from the set $\{0.00, 0.01, 0.02, 0.03\}$. Additionally, the system now implements one of the two OOL detectors, the Kohonen map OOL detector ($OOL_{SOM}$) or the PNN OOL detector ($OOL_{PNN}$). The mathematical framework formulation becomes:

**Objective Function**

$$\max_{x \in X} CA_{total}(x)$$

**Subject to: Performance Constraints**

$$\begin{aligned} E_{crit}(x) &< 0.1 && \text{upper bound on critical errors} \\ E_{ncrit}(x) &< 0.2 && \text{upper bound on non-critical errors} \\ P_{tp}(x) &> 0.85 && \text{lower bound on true positive rate} \end{aligned}$$

**Parameter Constraints**

$$\begin{aligned} \sum_{i=1}^{q} Q_i &= 1 && \text{use one of the } q \text{ quantile methods} \\ \sum_{i=1}^{w} W_i &= 1 && \text{use one of the } w \text{ wedge sizes} \\ \sum_{i=1}^{n} \theta_{1i} &= 1 && \text{use one of the } n \text{ noise thresholds} \\ \sum_{i=1}^{s} M_i &= 1 && \text{use one of the } s \text{ similarity metrics} \\ \sum_{i=1}^{f} F_i &= 1 && \text{use one of the } f \text{ fusion methods} \end{aligned}$$

**Non-Declaration Constraint**

$$\sum_{i=1}^{d} D_i = 1 \qquad \text{use one of the } d \text{ non-declaration threshold}$$

**OOL-Detector Constraint**

$$\sum_{i=1}^{g} G_i = 1 \qquad \text{use one of the } g \text{ OOL detectors}$$

With the entire composite classification system in place. We test the full capability of the system on the complete DCS dataset. As done in previuos research [3, 12, 34, 36], we not only implement a NDEC option and an OOL detector, but also allow for additional looks at a given target. By increasing the number of looks or observations the classification system has at each target, we should observe increased classification system performance. We note the results reported by Leap in Table 5.12, that showed increase in classifier performance for 2 looks and 5 looks while observing a degradation in OOL CA when using 10 observations. Leap concluded that the non-heterogenous nature of the data was the cause for this. To clarify, a brief explanation of the procedure is needed. When implementing the multiple target look methodology, target observations from subsequent aspect angles are combined with current observations then combined to produce a classifier output. For example, when observing a target at a given aspect angle, $\alpha$, additional looks would be made for the same target at aspect angles $\{\alpha + 1, \alpha + 2, ...\}$ in order to increase the likelihood of correct target identification. This can be accomplished by either: combining features at additonal looks, then inputing a combined feature set into the classifer; or by inputting the subsequent observations features sequentially and combining the outputs or comparison metrics of the classifier. For example, Leap [36] takes the latter approach. For each subseqeuent look at a given target, he computes the Mahalanobis distance between the target and each template class. Then, the average Mahalanobis distance over all looks is computed for each template class. These average distances are then used in the classification decsion. This was Leap's point for his conclusion about the non-heterogenous nature of the data. As the number of looks increased from 5 to 10, the features of the target class differed enough that the Mahalanobis distance between the target and it's true class will increase significantly. To illustrate this point, we show an example of two HRR profiles that differ by one degree of aspect angle in Figure 5.15. We see a significant change in the HRR profiles over a small shift in aspect angle, which can become more severe as the change in aspect angle increases.

In all previously presented experiments, we have demonstrated the capability of our classifier in both a 10-class forced decsion scenario and a 10-class with NDEC option scenario. In both of these cases, we demonstrate that optimal classifier performance is observed using the Global Model, in which all possible parameter settings and fusion rules are run prior to optimizing the system. This was compared to a Local Optimization Model, in which parameter settings were optimized prior to fusing. In both scenarios, we observed better results from the Global Model and tested the statistical significance of the conclusion. We have also independently tested the

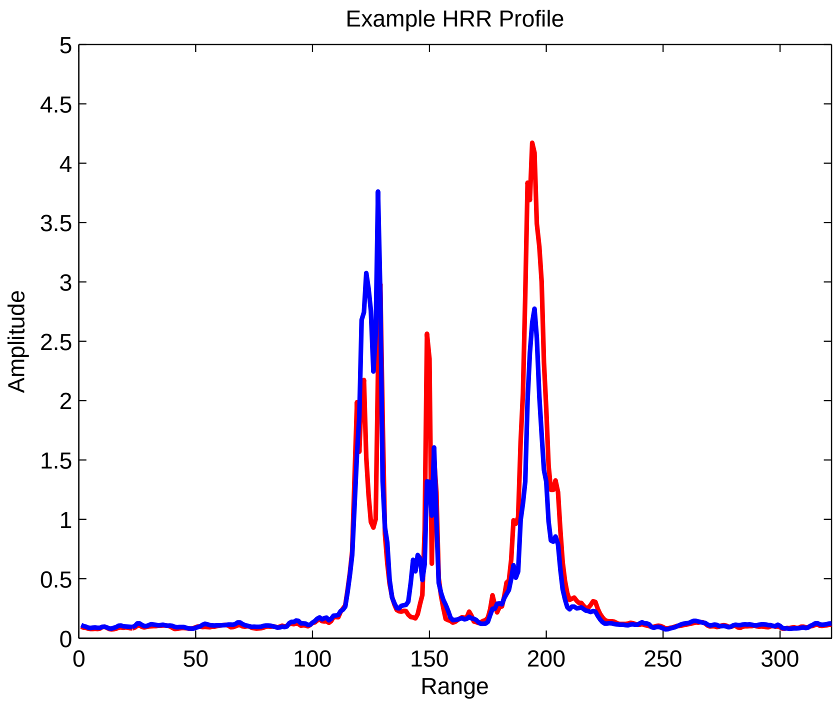

Figure 5.15  Example of Adjacent HRR Profile Shift.

capability of both the SOM and PNN OOL Detectors, with very favorable results under the most difficult of scenarios.

We now wish to test our classification system under a realistic operational scenario, with user defined constraints placed upon the classification system. Friend [12] used the optimization framework shown in Figure 5.16 for his tests on the DCS data with optimal results given in Table 5.15. We duplicate this effort and run our classification system under the Friend optimization framework.

We run a multiple look experiment using our composite classification system, under the same optimization framework as Friend. In order to make a comparision, we choose to optimize $CA_H$ under the framework. Recall, previous results we have

$\max_{x \in X} CA(x)$ maximizes total classification accuracy

Subject to:

Warfighter constraints

| | | |
|---|---|---|
| $E_{Crit} < 0.1$ | upper bound on critical errors |
| $E_{Ncrit} < 0.2$ | upper bound on non-critical errors |
| $P_{CA} > 0.85$ | lower bound on (total) classification accuracy |
| $P_{Dec} > 0.5$ | lower bound on declaration performance |
| $P_{OOL} > 0.35$ | lower bound on out-of-library performance |

Non-Declaration constraint

$\sum_{i=1}^{d} D_i = 1$ where $D_i = 1$ if $i$th non-dec method is used, $d$ is number of methods

$0 \le \theta_i \le 1$ decision threshold for the $i$th non-dec method

Figure 5.16 Friend OOL Framework [12].

reported were for $CA_{total}$. The optimal result across numbers of looks, along with the corresponding parameter settings are given in Table 5.17. As in previous experiments, the Global Model produces superior results to the Local Model. We list all optimal parameter settings for this experiment in Table 5.16

In comparing the results of our procedure listed in Table 5.17 to the results previuously reported by Friend listed in Table 5.15, we note the following improvements. First, our composite classification system has more solutions meeting all the constraints of the Friend Framework. This is true for both the Local Model and the Global Model, whose best result for each sensor/look combination is shown in Table 5.17. Second, the problem of degradation of OOL CA the Friend methodology suffered from is eliminated with our composite classifier. Recall Table 5.12 showed increase in classifier performance for 2 looks and 5 looks while observing a degradation in OOL CA when using 10 observations. Under our composite classification system, an increase in MOEs is observed for any classification accuracy. At the same time, both critical and non-critical error rates go down. Third, we note that although as we increase the number of looks, we observe an increase in our declaration rate. However, the Friend methodology has much higher declaration rates for the same number of looks. This is the obvious tradeoff. A lower declaration rate for higher classification accuracies. Finally, we observe that the combination of our independent OOL Detectors with the hybrid classifier does an outstanding job of correctly identifying IL and OOL targets, as evidenced by the $CA_{OOL}$ column of Table 5.17.

### *5.5.8 Extended Operating Conditions*

Our previous results have shown that the Hybrid Classification system improves results from previous research under nominal operating conditions (NOC). In nominal operating condidtions, the template, training and test data all come from the same data set, where the desired depression angle during data collection was between 6° and 8°. We now test our classification system under extended operating conditions (EOC). As previoulsy performed by Friend [12], in EOC experiments, a classifier is trained under nominal conditions and tested under extended conditions. In the case of EOC, the test data now comes from a seperate data set, where the desired depression angle during data collection was 10°.

Table 5.7 Hybrid Classifier Results: Non-Declaration Option.

| **Local Optimization Model** | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Parameters** | | | | | | **MOEs** | | | | | | | | |
| $Q$ | $\theta_1$ | $W$ | $M$ | $F$ | d | $CA_{total}$ | $LA_{total}$ | $CA_H$ | $LA_H$ | $CA_{FN}$ | $LA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ | DecRate |
| 5 | 0.5 | 7 | Euclidean | HH Data | 0.03 | 0.719 | 0.722 | 0.890 | .0.877 | 0.880 | 0.878 | 0.246 | 0.220 | 0.86 |
| 5 | 0.5 | 7 | Euclidean | VV Data | 0.03 | 0.737 | 0.744 | 0.925 | 0.868 | 0.859 | 0.896 | 0.171 | 0.225 | 0.87 |
| Fused | 0.5 | 7 | Euclidean | BEM | 0.03 | 0.783 | 0.792 | 0.950 | 0.907 | 0.902 | 0.902 | 0.074 | 0.139 | 0.80 |
| Fused | 0.5 | 7 | Euclidean | Borda | 0.03 | 0.703 | 0.720 | 0.926 | 0.848 | 0.833 | 0;833 | 0.120 | 0.176 | 0.977 |
| Fused | 0.5 | 7 | Euclidean | PNN | 0.03 | **0.801** | 0.804 | 0.923 | 0.933 | 0.935 | 0.935 | 0.071 | 0.123 | 0.761 |
| Fused | 0.5 | 7 | Euclidean | Bayes | 0.03 | 0.749 | 0.752 | 0.893 | 0.904 | 0.908 | 0.908 | 0.099 | 0.149 | 0.937 |
| **Global Optimization Model** | | | | | | | | | | | | | | |
| **Parameters** | | | | | | **MOEs** | | | | | | | | |
| $Q$ | $\theta_1$ | $W$ | $M$ | $F$ | d | $CA_{total}$ | $LA_{total}$ | $CA_H$ | $LA_H$ | $CA_{FN}$ | $LA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ | DecRate |
| 13 | 0.5 | 7 | Manhattan | BEM | 0.03 | **0.912** | 0.903 | 0.958 | 0.946 | 0.953 | 0.953 | 0.045 | 0.045 | 0.584 |
| 11 | 0.5 | 7 | Manhattan | Borda | 0.03 | 0.731 | 0.745 | 0.939 | 0.845 | 0.829 | 0.829 | 0.116 | 0.152 | 0.959 |
| 5 | 0.5 | 7 | Manhattan | PNN | 0.03 | 0.814 | 0.817 | 0.926 | 0.931 | 0.933 | 0.933 | 0.070 | 0.111 | 0.711 |
| 11 | 0.5 | 7 | Manhattan | Bayes | 0.03 | 0.750 | 0.751 | 0.887 | 0.888 | 0.890 | 0.890 | 0.112 | 0.134 | 0.992 |

Table 5.8 Hybrid Classifier Results: Non Dec Option, Model 1, PNN.

| $\mathbf{C} \downarrow \mathbf{L} \rightarrow$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NDEC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **237** | 14 | 2 | 0 | 2 | 9 | 1 | 1 | 5 | 11 | 78 |
| 2 | 7 | **263** | 2 | 9 | 3 | 2 | 1 | 0 | 0 | 1 | 72 |
| 3 | 0 | 5 | **206** | 2 | 25 | 3 | 1 | 6 | 2 | 0 | 110 |
| 4 | 0 | 3 | 15 | **149** | 15 | 3 | 4 | 23 | 2 | 0 | 146 |
| 5 | 1 | 3 | 35 | 9 | **159** | 4 | 3 | 11 | 0 | 0 | 135 |
| 6 | 15 | 9 | 3 | 2 | 2 | **231** | 1 | 2 | 8 | 21 | 66 |
| 7 | 2 | 0 | 1 | 1 | 0 | 0 | **207** | 16 | 5 | 2 | 126 |
| 8 | 0 | 2 | 3 | 5 | 7 | 1 | 23 | **176** | 3 | 0 | 140 |
| 9 | 10 | 1 | 5 | 1 | 3 | 3 | 19 | 9 | **207** | 5 | 97 |
| 10 | 12 | 0 | 1 | 1 | 1 | 11 | 0 | 0 | 2 | **261** | 71 |

Table 5.9 Hybrid Classifier Results: Non Dec Option, Model 2, BEM.

| $\mathbf{C} \downarrow \mathbf{L} \rightarrow$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | NDEC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | **233** | 11 | 2 | 1 | 2 | 10 | 0 | 1 | 10 | 2 | 88 |
| 2 | 6 | **261** | 0 | 4 | 4 | 0 | 2 | 0 | 0 | 0 | 83 |
| 3 | 0 | 3 | **149** | 0 | 13 | 0 | 0 | 3 | 0 | 0 | 192 |
| 4 | 0 | 0 | 5 | **128** | 1 | 0 | 3 | 3 | 1 | 0 | 219 |
| 5 | 0 | 1 | 5 | 9 | **104** | 0 | 0 | 3 | 3 | 0 | 244 |
| 6 | 2 | 9 | 3 | 4 | 0 | **238** | 0 | 1 | 5 | 12 | 86 |
| 7 | 0 | 0 | 1 | 1 | 0 | 1 | **202** | 7 | 0 | 0 | 148 |
| 8 | 0 | 0 | 3 | 0 | 2 | 0 | 0 | **134** | 0 | 0 | 221 |
| 9 | 6 | 1 | 0 | 0 | 5 | 0 | 6 | 0 | **234** | 0 | 108 |
| 10 | 11 | 2 | 0 | 0 | 3 | 5 | 0 | 0 | 0 | **229** | 110 |

Table 5.10 Hybrid Classifier: NDEC Parameter Settings.

| Model | $Q$ | $\theta_1$ | $W$ | $M$ | $F$ |
|---|---|---|---|---|---|
| Local | 5 | 0.5 | 7 | Euclidean | BEM |
| Global | 13 | 0.5 | 7 | Manhattan | BEM |

Table 5.11 Out-of-Library Targets with Descriptions and Characteristics.

| Type | Target Description | Tracks | Wheels | Gun |
|---|---|---|---|---|
| SA-8 TZM | SA-8 Reload Vehicle | N | 6 | N |
| BMP-1 | Tank w/small turret | Y | 0 | Y |
| BTR-70 | 8-wheeled transport | N | 8 | N |
| SA-13 | Turret SAMs | Y | 0 | N |
| SA-8 TEL | Integrated Radar Exposed SAMs | N | 6 | N |

Table 5.12 Results Summary for Friend Feature Vector Methodology

| Senor | Looks | TPR | IL CA | OOL CA | Dec |
|---|---|---|---|---|---|
| HH | 1 | 0.98 | 0.95 | 0.70 | 0.23 |
| VV | 1 | 0.97 | 0.96 | 0.70 | 0.24 |
| HH | 2 | 0.97 | 0.94 | 0.70 | 1 |
| VV | 2 | 0.96 | 0.95 | 0.72 | 1 |
| HH | 5 | 0.99 | 0.97 | 0.68 | 1 |
| VV | 5 | 0.97 | 0.96 | 0.72 | 1 |
| HH | 10 | 0.99 | 0.98 | 0.59 | 1 |
| VV | 10 | 0.97 | 0.98 | 0.62 | 1 |

Table 5.13 Results Summary for SOM OOL Detector Experiment.

| Result ↓ True Class → | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| TP | 56 | 35 | 37 | 65 | 58 | 47 | 43 | 56 | 74 | 54 |
| FP | 304 | 325 | 323 | 295 | 302 | 313 | 317 | 304 | 286 | 306 |
| TN | 1500 | 1620 | 1504 | 1446 | 1436 | 1391 | 1461 | 1522 | 1395 | 1425 |
| FN | 300 | 180 | 296 | 354 | 364 | 409 | 339 | 278 | 405 | 375 |

Table 5.14 Results Summary for PNN OOL Detector Experiment.

| Result ↓ True Class → | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| TP | 44 | 41 | 35 | 10 | 36 | 44 | 7 | 51 | 30 | 24 |
| FP | 316 | 319 | 325 | 350 | 324 | 316 | 353 | 309 | 330 | 336 |
| TN | 1656 | 1684 | 1737 | 1778 | 1753 | 1625 | 1795 | 1737 | 1684 | 1646 |
| FN | 144 | 116 | 63 | 22 | 47 | 175 | 5 | 63 | 116 | 154 |

Table 5.15 Friend OOL Optimization Results.

| **Parameters** | | **MOEs** | | | | | |
|---|---|---|---|---|---|---|---|
| **Looks** | **Sensor** | $CA_H$ | $CA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ | $CA_{OOL}$ | $Dec$ |
| 2 | HH | 0.9454 | 0.8725 | 0.0030 | 0.0663 | 0.3530 | 0.5585 |
| 2 | VV | 0.9285 | 0.8661 | 0.0125 | 0.0672 | 0.3919 | 0.7713 |
| 5 | HH | 0.9472 | 0.8782 | 0.0157 | 0.0510 | 0.3535 | 0.8828 |
| 5 | VV | 0.9255 | 0.8370 | 0.0317 | 0.0524 | 0.3530 | 0.9583 |
| 10 | HH | 0.9537 | 0.8744 | 0.0270 | 0.0322 | 0.4580 | 0.9880 |
| 10 | VV | 0.9465 | 0.8308 | 0.0322 | 0.0448 | 0.4580 | 0.9880 |
| 2 | Mean | 0.9233 | 0.7277 | 0.0572 | 0.0633 | 0.4039 | 0.9943 |
| 5 | Mean | 0.9143 | 0.7559 | 0.0494 | 0.0622 | 0.4921 | 0.9889 |
| 10 | Mean | 0.9339 | 0.8249 | 0.0402 | 0.0452 | 0.4971 | 0.9759 |

Table 5.16 Composite Classifier System Optimal Parameter Settings.

| Model | OOL Detector | $Q$ | $\theta_1$ | $W$ | $M$ | $F$ |
|---|---|---|---|---|---|---|
| Global | PNN | 11 | 0.5 | 7 | Manhattan | PNN |

Table 5.17 Results Summary for Composite Classifier OOL Optimization Experiment NOC.

| **Parameters** | | **MOEs** | | | | | |
|---|---|---|---|---|---|---|---|
| Looks | Sensor | $CA_H$ | $CA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ | $CA_{OOL}$ | $Dec$ |
| 1 | HH | 0.927 | 0.912 | 0.081 | 0.099 | 0.659 | 0.554 |
| 1 | VV | 0.953 | 0.945 | 0.0511 | 0.0846 | 0.609 | 0.571 |
| 1 | BEM | 0.9554 | 0.961 | 0.062 | 0.0663 | 0.641 | 0.561 |
| 1 | Borda | 0.946 | 0.945 | 0.082 | 0.0663 | 0.650 | 0.559 |
| 1 | PNN | 0.9454 | 0.952 | 0.075 | 0.067 | 0.661 | 0.558 |
| 1 | Bayes | 0.9454 | 0.922 | 0.079 | 0.066 | 0.649 | 0.562 |
| 2 | HH | 0.929 | 0.942 | 0.064 | 0.064 | 0.690 | 0.550 |
| 2 | VV | 0.954 | 0.959 | 0.043 | 0.060 | 0.632 | 0.559 |
| 2 | BEM | 0.961 | 0.973 | 0.065 | 0.0663 | 0.685 | 0.551 |
| 2 | Borda | 0.955 | 0.952 | 0.048 | 0.0663 | 0.692 | 0.563 |
| 2 | PNN | 0.959 | 0.962 | 0.052 | 0.0663 | 0.698 | 0.560 |
| 2 | Bayes | 0.933 | 0.958 | 0.066 | 0.0663 | 0.655 | 0.5585 |
| 5 | HH | 0.954 | 0.959 | 0.013 | 0.004 | 0.845 | 0.585 |
| 5 | VV | 0.999 | 0.997 | 0.002 | 0.008 | 0.833 | 0.587 |
| 5 | BEM | 0.973 | 0.966 | 0.010 | 0.007 | 0.852 | 0.588 |
| 5 | Borda | 0.966 | 0.962 | 0.008 | 0.009 | 0.855 | 0.585 |
| 5 | PNN | 0.972 | 0.967 | 0.009 | 0.008 | 0.862 | 0.588 |
| 5 | Bayes | 0.955 | 0.954 | 0.011 | 0.008 | 0.851 | 0.583 |
| 10 | HH | 1.00 | 1.00 | 0 | 0 | 0.947 | 0.688 |
| 10 | VV | 1.00 | 1.00 | 0 | 0 | 0.926 | 0.726 |
| 10 | BEM | 1.00 | 1.00 | 0 | 0 | 0.946 | 0.698 |
| 10 | Borda | 1.00 | 1.00 | 0 | 0 | 0.954 | 0.712 |
| 10 | PNN | 1.00 | 1.00 | 0 | 0 | 0.954 | 0.723 |
| 10 | Bayes | 1.00 | 1.00 | 0 | 0 | 0.943 | 0.720 |

Table 5.18 Friend OOL Optimization Results EOC.

| **Parameters** | | **MOEs** | | | | | |
|---|---|---|---|---|---|---|---|
| **Looks** | **Sensor** | $CA_H$ | $CA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ | $CA_{OOL}$ | $Dec$ |
| 2 | HH | 0.8579 | 0.9270 | 0.0066 | 0.0655 | 0.2026 | 0.5033 |
| 2 | VV | 0.8875 | 0.9167 | 0.0245 | 0.0515 | 0.2675 | 0.5069 |
| 5 | HH | 0.8516 | 0.8891 | 0.0035 | 0.0870 | 0.2552 | 0.5319 |
| 5 | VV | 0.8502 | 0.9347 | 0.0199 | 0.0591 | 0.2779 | 0.5017 |
| 10 | HH | 0.8782 | 0.8046 | 0.0089 | 0.1018 | 0.3505 | 0.52220 |
| 10 | VV | 0.9036 | 0.8559 | 0.0206 | 0.0695 | 0.3555 | 0.6098 |
| 2 | Mean | 0.8501 | 0.8202 | 0.0379 | 0.0807 | 0.2941 | 0.5624 |
| 5 | Mean | 0.8985 | 0.8085 | 0.0310 | 0.0719 | 0.3512 | 0.5794 |
| 10 | Mean | 0.9328 | 0.7750 | 0.0385 | 0.0676 | 0.3554 | 0.6822 |

Table 5.19 Results Summary for Composite Classifier OOL Optimization Experiment EOC.

| **Parameters** | | **MOEs** | | | | | |
|---|---|---|---|---|---|---|---|
| Looks | Sensor | $CA_H$ | $CA_{FN}$ | $E_{crit}$ | $E_{ncrit}$ | $CA_{OOL}$ | $Dec$ |
| 1 | HH | 0.836 | 0.827 | 0.091 | 0.81 | 0.525 | 0.544 |
| 1 | VV | 0.847 | 0.836 | 0.089 | 0.084 | 0.519 | 0.562 |
| 1 | BEM | 0.850 | 0.848 | 0.089 | 0.082 | 0.524 | 0.552 |
| 1 | Borda | 0.849 | 0.852 | 0.086 | 0.088 | 0.522 | 0.548 |
| 1 | PNN | 0.855 | 0.851 | 0.081 | 0.079 | 0.528 | 0.549 |
| 1 | Bayes | 0.852 | 0.849 | 0.087 | 0.079 | 0.530 | 0.544 |
| 2 | HH | 0.909 | 0.912 | 0.073 | 0.076 | 0.590 | 0.571 |
| 2 | VV | 0.921 | 0.923 | 0.064 | 0.078 | 0.572 | 0.578 |
| 2 | BEM | 0.912 | 0.918 | 0.065 | 0.076 | 0.558 | 0.581 |
| 2 | Borda | 0.919 | 0.921 | 0.058 | 0.066 | 0.592 | 0.573 |
| 2 | PNN | 0.936 | 0.925 | 0.072 | 0.066 | 0.589 | 0.568 |
| 2 | Bayes | 0.923 | 0.918 | 0.066 | 0.0663 | 0.655 | 0.5585 |
| 5 | HH | 0.944 | 0.939 | 0.013 | 0.004 | 0.765 | 0.683 |
| 5 | VV | 0.949 | 0.947 | 0.012 | 0.017 | 0.773 | 0.682 |
| 5 | BEM | 0.953 | 0.956 | 0.019 | 0.018 | 0.785 | 0.689 |
| 5 | Borda | 0.964 | 0.962 | 0.013 | 0.019 | 0.785 | 0.684 |
| 5 | PNN | 0.952 | 0.967 | 0.012 | 0.017 | 0.782 | 0.698 |
| 5 | Bayes | 0.955 | 0.954 | 0.011 | 0.021 | 0.768 | 0.689 |
| 10 | HH | 0.98 | 0.98 | 0 | 0 | 0.833 | 0.677 |
| 10 | VV | 0.99 | 0.98 | 0 | 0 | 0.828 | 0.706 |
| 10 | BEM | 0.99 | 0.98 | 0 | 0 | 0.855 | 0.698 |
| 10 | Borda | 0.99 | 0.98 | 0 | 0 | 0.861 | 0.702 |
| 10 | PNN | 1.00 | 0.99 | 0 | 0 | 0.884 | 0.710 |
| 10 | Bayes | 0.99 | 0.99 | 0 | 0 | 0.863 | 0.715 |

# 6. Contributions and Future Research

This chapter provides a summary of the contributions made to the fields of pattern recogntion, automatic target recognition and operations research by this research.

## *6.1 Research Contributions*

### *6.1.1 Hybrid Template-Based Classifier Development*

This research develops a combined hybrid template-based classification system that operates effectively across two seperate application areas. The development of the classification system includes the exploration of feature extraction, representation, similiarity measures, classification decisions and fusion techniques for the purpose of achieving optimal classifier performance. A major contribution made during the development is the ability of the classifier system to operate in different application domains while mainitaining both superior identification rate and low computational time.

### *6.1.2 Combinded Classifier Development*

We explore various fusion schemes and techniques in order to boost overall classifier system performance. A key technique we explore is the order of operations for optimzation and fusing. Under one scenario, called the Local Model, independent classifiers are optimized prior to fusion. Under the second scenario, called the

Global Model, the entire parameter space across all classifiers are explored prior to optimization. This second scenario is shown to produce superior results under intitial exploration. Furthermore, these models are then replicated such that tests for statistical significance can be performed to validate the intitial model comparison conclusions.

### *6.1.3 Improvements to existing NDEC and OOL methodologies*

Using a very simple NDEC methodology, we seek the ability to quickly identify situations for which the classification system is unable to distinguish which labeling assignment to make. Under these conditions, comparisons between a test exemplar and at least two target templates lead to possible label assignments, with the classification system unable to properly or quickly decide between the available choices due to the close resemblance of the labelling candidates. By simplifying the calculations to a comparison between candidates with the overall range of distance and/or similarity measures, the classifier is able to make an accurate and timely classification decision.

### *6.1.4 OOL Methodology Development*

We develop OOL detectors that operate independent of the classifier through the use of two separate artificial neural networks: a self-organizing map (SOM) and probabalistic neural network (PNN). The OOL Detectors we develop can operate on any features taken from the original data. We can use as many or as few as we choose without having to make dramatic changes to the overall algorithmic structure of either OOL Detector. Moreover, the OOL Detectors both fall under the category of Blind Mehods. A *Blind* Method is based solely on in-library training data without any knowledge of test data class membership. OOL criteria is based entirely on in-library training data and the characteristics of a test exemplar.

We have thus developed two independent OOL Detectors that operate independently of the classifier from which they receive inputs. Moreover, they do not create their decision thresholds from any information other than training data. Thus, our OOL Detectors operate under the most difficult of conditions, while producing very favorable results.

### *6.1.5 Mathematical Framework*

We provide the mathematical framework for our compbined classification system. This framework encompasses the implementation of methodology for the case of OOL targets as well as the NDEC option. We clearly define the mathematics of our representation scheme, parameter space and similarity measures and distance metrics which are used to produce classifier outputs or labels. Finally, we formulate a mixed variable optimization problem as well as various methods of evaluation in seeking to improve classification system performance.

## *6.2 Future Research*

During the course of research, different avenues present themselves, which due to time limitations, are left as areas of future research. This section suggests items of future work.

### *6.2.1 Feature Selection*

Most research on the identification of targets within the HRR profile problems have considered features extracted from the amplitudes of HRR profiles measured across the range of the profile. Exploration of other features, such as occurences of different amplitudes or the measured horizontal distance of a certain portion of the profile have yet to be explored. For example, using the noise threshold of this research, a horizontal range distance between the first and last amplitudes which exceed the noise threshold could be used as either a seperate feature space or used to augment pre-existing features.

### *6.2.2 Robustness*

In all of our experiments, classification system performance was evaluated based solely on maximizing performance, such as overall classification accuracy. Another avenue for evaluating classification system performance is that of robustness. Under this type of performance, a more consistent classification system can be developed by seeking to find optimal parameters which reduce variance of certain performances, rather than the maximization problem used here.

### *6.2.3 Generalzied OOL Detector*

The OOL Detectors in this research were two specific ANNs: the SOM or the PNN. Future research could explore other techniques, whether those techniques are other ANNs or different methodologies. Further exploration of the use of all in-library targets such as creating sub-division of in-library target by description or function could also be useful in creating an OOL Detector.

### *6.2.4 Generalized Composite Classifier*

This research was able to successfully implement two independent classifiers with one of two OOL Detectors. Further exploration of other classifiers, OOL detectors and fusion methods using various methods could prove to be fruitful. As an example, the research we present operates with all classifiers and OOL Detectors operating in parallel. Different hierarchical schemes or interactions between the classifiers and/or OOL Detectors could lead to improved performance.

# Appendix A. List of Abbreviations and Terms

**AFRL** Air Force Research Laboratory

**ATR** Automatic Target Recognition

**BEM** Basic Ensemble Method

**CA** Classification Accuracy

**CID** Combat Identification

**DCS** Data Collection System

**Dec** Declarations; the percentage of exemplars labeled by a classifier

**DOE** Design of Experiment

$E_{crit}$ Probability of a critical error

$E_{ncrit}$ Probability of a non-critical error

**FFT** Fast Fourier Transform

**FVPR** Feature Vector Pattern Recognition

**FN** Friendly or Neutral Target or Class

**H** Hostile Target Class, includes both TOD and OH

**HH** Horizontally polarized radar transmit and receive

**HPR** Hybrid Pattern Recognition

**HRR** High Range Resolution

**LA** Label Accuracy

**MCS** Multiple Classifier System

**MOE** Measure of Effectiveness

**MSTAR** Moving and Stationary Target Acquisition and Recognition

**MVP** Mixed Variable Programming

**NDEC** Non-Declarations; Percentage of exemplars not labeled by a classifier

**NIST** National Institiute Standards and Technology

**OCR** Optical Character Recognition

**OH** Other Hostile

**OOL** Out of Library

**PNN** Probabilistic Neural Network

**RBF** Radial Basis Function

**ROC** Receiver Operating Characteristic

**SAR** Synthetic Apperture Radar

**TOD** Target of the Day

**VV** Vertically polarized radar transmit and receive

# Bibliography

1. (2007). Bayes net toolbox for matlab. http://www.cs.ubc.ca/ murphyk/Software/BNT/bnt.html.

2. Abe, N. and Mamitsuka, H. (1997). Predicting protein secondary structure using stochastic tree grammars. *Machine Learning*, **29**, 275.

3. Albrecht, T. (2005). *Combat Identification with Sequential Observations, Rejection Option and Out-of-Library Targets*. Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

4. Albrecht, T. (2005). Introduction to fusion of multiple classifiers. AFIT OPER 786 Guest Lecture, Wright-Patteron AFB, OH.

5. Bhatnagar, R., Williams, R., and Tennety, V. (2000). Syntactic pattern recognition of HRR signatures. In *Proceedings of SPIE's 2000 Conference on Algorithms for Synthetic Aperture Radar Imagery*, volume 4053. 452–466.

6. Breiman, L. (1996). Bagging predictors. *Machine Learning*, **24**, 123–140.

7. Bunke, H. (1990). *Syntactic and Structural Pattern Recognition Theory and Applications edited by H. Bunke and A. Sanfeliu*, chapter 11: Hybrid Pattern Recognition Methods. World Scientific Publishing.

8. Chow, C. (1970). On optimum rejection error and reject tradeoff. *IEEE Transactions on Information Theory*, **16**, 41–46.

9. Connell, S. and Jain, A. (2001). Template-based online character recognition. *Pattern Recognition*, **34**, 1–14.

10. DeWitt, M. (1992). *High range resolution radar target identification using the prony model and hidden markov models*. Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

11. Duda, R., Hart, P., and Stork, D. (2001). *Pattern Classification*. John Wiley and Sons, Inc., Hoboken, NJ, second edition.

12. Friend, M. (2007). *Combat Identification with Synthetic Aperture Radar, Hidden Markov Models and Information Theory*. Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

13. Fu, K.-S. (1981). *Pattern Recognition Theory and Applications*, chapter Hybrid Approaches to Pattern Recognition. Reidel Dordrecht.

14. Fu, K.-S. (1982). *Syntactic Pattern Recognition and Applications*. Prentice-Hall, Englewood Cliffs, NJ, first edition.

15. Fumera, G., Roli, F., and Giorgio, G. (2000). Reject option with multiple thresholds. *Pattern Recognition*, **33**, 2099–2101.

16. Gallian, J. (1994). *Contemporary Abstract Algebra.* D.C. Heath and Company, Lexington, MA, third edition.

17. Govindan, V. (1990). Character recognition: A review. *Pattern Recognition*, **23**, 671–683.

18. Gusfield, D. (1997). *Algorithms on strings, trees and strings.* Cambridge University Press, New York, NY, first edition.

19. Hall, D. L. and LLinas, J. (1997). An introduction to multisensor data fusion. In *Proceedings of the IEEE*, volume 85. 1.

20. Hebert, A. J. (2004). Building battlespace awareness. *Air Force Magazine Online*, **9**, 66–71.

21. Heckerman, D. (1995). A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Corporation, Microsoft Research, Advanced Technology Division.

22. Ho, T. K. (1995). Random decision forests. *Document Analysis and Recognition, International Conference on*, **1**, 278.

23. Ho, T. K., Hull, J. J., and Srihari, S. N. (1994). Decision combination in multiple classifier systems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **16**, 66–75.

24. Ho, T. K., Lee, E. T., and Ho, T.-T. (1987). Syntactic approach to image analysis. In *Proceedings of ACM Computer Science Conference*, volume 395.

25. Jain, A. K., Duin, R. P., and Mao, J. (2000). Statistical pattern recognition: A review. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, **22**, 4–37.

26. Jaro, M. (1995). Probabilistic linkage of large public health data file. *Statistics in Medicine*, **14**, 491–498.

27. Jung, K., Gannoun, A., Sitek, B., Apostolov, O., Schramm, A., Meyer, H., Stuhler, K., and Urfer, W. (2006). Statistical evaluation of methods for the analysis of dynamic protein expression data from a tumor study. *REVSTAT Statisical Journal*, **4**, 67–80.

28. Kimura, F. and Shridhar, M. (1991). Handwritten numerical recognition based on multiple algorithms. *Pattern Recognition*, **24**, 969–983.

29. Kittler, J., Hatef, M., Duin, R., and Matas, J. (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Inteligence*, **20**, 226–239.

30. Klepko, R. (1991). Classification of SAR ship images with the aid of a syntactic pattern recognition algorithm. *NASA STI/Recon Technical Report N*, **92**, 27695–+.

31. Kohonen, T. (1997). *Self-organizing Maps.* Springer, Berlin,New York, second edition.

32. Koudelka, M. L., Richards, J. A., and Koch, M. W. (2007). Multinomial pattern matching for high range resolution radar profiles. In *Proceedings of SPIE*, volume 6568. 65680V.

33. Kuroda, K., Harada, K., and Hagwara, M. (1997). Large scale on-line handwritten chinese character recognition using improved syntactic pattern recognition. In *1997 IEEE Conference on Systems, Man and Cybernetics*, volume 5. 4530–4535.

34. Laine, T. (2005). *Optimization of Automatic Target Recogntion with a Reject Option Using Fusion and Correlated Sensor Data.* Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

35. Leap, N. (2004). *An Investigation of the Effects of Correlation, Autocorrlation and Sample Size in Classifier Fusion.* Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

36. Leap, N. (2008). *A Confidence Paradigm for Classification Systems.* Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

37. Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE*, **86**, 2278–2324.

38. Lee, E. (1979). Space object surveillance and identification. *Policy Analysis and Information Systems*, **3**, 179–185.

39. Liu, C.-L., Jaeger, S., and Nakagawa, M. (2004). Online recognition of chinese characters: The state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **26**, 198–213.

40. Liu, C.-L., Nakashima, K., Sako, H., and Fujisawa, H. (2003). Handwritten digit recogntion: benchmarking of state of the art techniques. *Pattern Recognition*, **36**, 2271–2285.

41. Looney, C. G. (1997). *Pattern Recognition Using Neural Networks.* Oxford University, New York, NY.

42. MacDonald, A. (1999). *Classification of high range resolution radar returns using hidden markov and guassian mixture models.* Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

43. Mantas, J. (1986). An overview of character recognition methodologies. *Pattern Recognition*, **19**, 425–429.

44. Meyer, G. (2003). *Classification of Radar Targets using Invariant Features*. Ph.D. thesis, Air Force Institute of Technology, Wright-Patterson AFB OH.

45. Mitchell, R. and Westerkamp, J. (1999). Robust statistical feature based aircraft identification. *IEEE Transactions on Aerospace and Electronic Systems*, **35**(3), 1077–1094.

46. Mitchell, R. A. (1997). *Robust High Range Resolution Radar Target Identification using Statistical Feature Based Classifier with Feature Level Fusion*. Ph.D. thesis, University of Dayton, Dayton OH.

47. Moghaddam, B. and Petland, A. (1997). Probabilistic visual learning for object representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **19**, 696–710.

48. Nadler, M. and Smith, E. (1993). *Pattern Recognition Engineering*. John Wiley and Sons, Inc., Hoboken, NJ, first edition.

49. Ogiela, M. R. and Tadeusiewicz, R. (2000). Application of syntactic methods of pattern recognition for data mining and knowledge discovery in medicine. In *Proceedings of SPIE*, volume 4057. 308–318.

50. Oliver, C. and Quegan, S. (2004). *Understanding Synthetic Aperture Radar Images*. Artech House, Raleigh, NC, first edition.

51. Perrone, M. P. and Cooper, L. N. (1993). When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Neural Networks for Speech and Image Processing*. Chapman-Hall, 126–142.

52. Ramamoorthy, L. and Casasent, D. Classification and rejection of MSTAR data.

53. Richards, J. and Bray, B. K. An informative confidence metric in ATR.

54. Roli, F. (2003). Fusion of multiple pattern classifiers. 8th National Conference on the Italian Association of Arificial Intelligence, Pisa IT.

55. Roli, F. and Giacinto, G. (2002). *Hybrid Methods in Pattern Recognition by H. Bunke and A. Kandel*, chapter 8: Design of Multiple Classifier Systems. World Scientific Publishing.

56. Ross, T. and Minardi, M. (2004). Discrimination and confidence error in detector reported scores. In *Proceedings of SPIE: Algorithms for Synthetic Aperture Radar Imagery XI*, volume 5427.

57. Sadowski, C. (2008). Combat identification. AFIT OPER 786 Guest Lecture, Wright-Patterson AFB, OH.

58. Sakakibara, Y., Brown, M., Hughey, R., Mian, I., Sjolander, K., C.Underwood, R., and Haussler*, D. (July 1994). Stochastic context-free grammars for t rna modeling. Technical report, University of California and Fujitsu Laboratories.

59. Shaw, A. K., Vashist, R., and Williams, R. (2000). Iirr-.atr using eigen-templates with noisy observations in unknown target scenario. In *Proceedings of SPIE*, volume 4053. 467–478.

60. Teow, L.-N. and Loe, K. (2002). Robust vision-based features and classification schemes for off-line handwritten digit recognition. *Pattern Recognition*, **35**, 2355–2364.

61. Trahanias, P. and Skordalakis, E. (1990). Syntactic pattern recognition of the ecg. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **12**, 648–657.

62. Turnbaugh, M. and Bauer, K. (2008). HRR signature classification using a hybrid composite classification syste. In *Proceedings of SPIE*. 1–12.

63. Turnbaugh, M., Bauer, K., Oxley, M., and Miller, J. (2008). HRR signature classification using syntactic pattern recognition. In *IEEE Aerospace Conference*. 1–9.

64. United States Air Force (1998). *USAF Intelligence Targeting Guide, AFPAM 14-210.*

65. United States Air Force (1999). *Intelligence, Surveillance, and Reconnaissance Operations, AFDD 2-5.2.*

66. United States Air Force (2000). *Air Warfare,Air Force Doctrine Document 2-1,.*

67. Wanas, N. M. and Kamel, M. S. (2001). Decision fusion in neural network ensembles. In *Proceedings of International Joint Conference on Neural Networks*, volume 4. 2952–2957.

68. Wasserman, P. D. (1993). *Advanced Methods in Neural Computing.* Van Nostrand Reinhold, New York, NY, first edition.

69. Webb, A. (2002). *Statistical Pattern Recognition.* John Wiley and Sons, Inc., Hoboken, NJ, second edition.

70. Welch, P. (1983). *The Statistical Analysis of Simulation Results.* Academic Press, Palisades, NY, first edition.

71. Williams, R., Gross, D., Palomino, A., Westerkamp, J., and Wardell, D. (1998). 1d HRR data analysis and atr assessment. In *Proceedings of SPIE*, volume 3370. 588–599.

72. Williams, R., Gross, D., Palomino, A., Westerkamp, J., and Wardell, D. (2000). Automatic target recognition of time critical moving targets using 1d high range resolution radar. *IEEE AES Systems Magazine*, 37–43.

73. Williams, R., Westerkamp, J., Gross, D., Palomino, A., Kaufman, T., and Fister, T. (1999). Analysis of a 1-d hrr moving target atr. In *Proceedings of SPIE*, volume 3721. 413–424.

74. Winkler, W. E. The state of record linkage and current research problems. Technical report, United States Internal Revenue Service.

75. Xu, L., Krzyzak, A., and Suen, C. Y. (1992). Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, **22**, 418–435.

76. Zajic, T., Rago, C., Mahler, R., Huff, M., and Noviskey, M. (2001). Joint tracking, pose estimation and target recognition using HRR and track data: new results. In *Proceedings of SPIE*, volume 4380. 196–206.

# Vita

Captain Micheal A. Turnbaugh graduated from Flour Bluff High School in Corpus Christi, Texas. He enlisted in the United States Air Force in 1990 and served an Emergency Medical Technician at Air University Hospital, Maxwell AFB and an Aeromedical Evacuation Technician at the 1st Aeromedical Evacuation Squadron, Pope AFB.

Captain Turnbaugh earned a Bachelor of Science Degree in Mathematics from Fayetteville State Univerisity in 1998 and a Master of Science Degree in Mathematics from the University of North Dakota in 2000. He was commissioned as a Distiguished Graduate of Officer Training School in August 2000.

Captain Turnbaugh was then assigned to Eglin AFB, FL where he served as a Weapons and Tactics Analyst in the 86th Fighter Weapons Squadron, 53rd Weapons Evaluation Group, 53rd Wing. In July 2003, he was assigned to the United States Air Force Academy as an Instructor and later Assistant Professor in the Department of Mathematical Sciences. In September 2005, he entered the Graduate School of Engineering and Management, Air Force Institute of Technology. Upon graduation, he will be assigned to HQ Air Mobility Command, Scott AFB, IL.

| REPORT DOCUMENTATION PAGE | *Form Approved* *OMB No. 074-0188* |
|---|---|

| | | |
|---|---|---|
| **1. REPORT DATE** *(DD-MM-YYYY)* 02-05-2009 | **2. REPORT TYPE** **Doctoral Dissertation** | **3. DATES COVERED** *(From – To)* Sept 2005 – Jan 2009 |

| | |
|---|---|
| **4. TITLE AND SUBTITLE** A HYBRID TEMPLATED-BASED COMPOSITE CLASSIFICATION SYSTEM | **5a. CONTRACT NUMBER** |
| | **5b. GRANT NUMBER** |
| | **5c. PROGRAM ELEMENT NUMBER** |
| **6. AUTHOR(S)** Turnbaugh, Michael A., Captain, USAF | **5d. PROJECT NUMBER** |
| | **5e. TASK NUMBER** |
| | **5f. WORK UNIT NUMBER** |
| **7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)** Air Force Institute of Technology<br>Graduate School of Engineering and Management (AFIT/EN)<br>2950 Hobson Street, Building 642<br>WPAFB OH 45433-7765 | **8. PERFORMING ORGANIZATION REPORT NUMBER** AFIT/DS/ENS/08-04 |
| **9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** AFOSR/NM, Dr. Fariba Fahroo<br>Suite 325 Rm 3112<br>875 Randolph<br>Arlington VA 22203-1768<br>ACC/DRSA, Mr. Charles Sadowski<br>216 Hunting Ave, Rm 105<br>Langley AFB VA 23665-2777 | **10. SPONSOR/MONITOR'S ACRONYM(S)** **AFOSR/NM, ACC/DRSA** |
| | **11. SPONSOR/MONITOR'S REPORT NUMBER(S)** |

**12. DISTRIBUTION/AVAILABILITY STATEMENT**

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT**

An automatic target classification system contains a classifier which reads a feature as an input and outputs a class label. Typically, the feature is a vector of real numbers. Other features can be non-numeric, such as a string of symbols or alphabets. One method of improving the performance of an automatic classification system is through combining two or more independent classifiers that are complementary in nature. This research proposes a design for a hybrid composite classification system, which exploits quantized integer valued features with a template matching classification scheme. This composite classification system is made up of independent classification systems which are combined over various fusion methods within a mathematical framework to produce optimal classifier performance. These two independent classification systems, which receive input from two separate sensors, are then combined over various fusion methods for the purpose of target identification. By using these two separate classifiers, we explore conditions that allow the two techniques to be complementary in nature, thus improving the overall performance of the classification system. We examine various fusion techniques, in search of the technique that generates the best results. We investigate different parameter spaces and fusion rules on example problems to demonstrate our classification system. Our examples consider various application areas to help further demonstrate the utility of our classifier. Optimal classifier performance is obtained using a mathematical framework, which takes into account decision variables based on decision-maker preferences and/or engineering specifications, depending upon the classification problem at hand.

**15. SUBJECT TERMS**

Hybrid Classifiers, Template Classifiers, Classification, Automatic Target Recognition, Fusion, Out-of-Library

| **16. SECURITY CLASSIFICATION OF:** | | | **17. LIMITATION OF ABSTRACT** | **18. NUMBER OF PAGES** | **19a. NAME OF RESPONSIBLE PERSON** Dr. Kenneth Bauer, Jr. (AFIT/ENS) |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | **19b. TELEPHONE NUMBER** *(Include area code)* |
| U | U | U | UU | 187 | (937) 255-6565, ext 4328 |

**Standard Form 298 (Rev. 8-98)**
Prescribed by ANSI Std. Z39-18